

Novell SUSE® Linux Enterprise Server

10

February 1, 2007

STORAGE ADMINISTRATION GUIDE
FOR EVMS

www.novell.com



Novell®

Legal Notices

Novell, Inc., makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc., reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc., makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc., reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. Please refer to www.novell.com/info/exports/ for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Copyright © 2006 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

Novell, Inc., has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.novell.com/company/legal/patents/> and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

For a list of Novell trademarks, see the [Novell Trademark and Service Mark list \(http://www.novell.com/company/legal/trademarks/tmlist.html\)](http://www.novell.com/company/legal/trademarks/tmlist.html).

Third-Party Materials

All third-party trademarks and copyrights are the property of their respective owners.

Some content in this document is copied, distributed, and/or modified from the following document under the terms specified in the document's license.

EVMS User Guide, January 18, 2005

Copyright © 2005 IBM

License Information

This document may be reproduced or distributed in any form without prior permission provided the copyright notice is retained on all copies. Modified versions of this document may be freely distributed provided that they are clearly identified as such, and this copyright is included intact.

Contents

About This Guide	9
1 Overview of EVMS	11
1.1 Benefits of EVMS	11
1.2 Plug-In Layers	11
1.3 File Systems Support	12
1.4 Terminology	12
1.5 Location of Device Nodes for EVMS Storage Objects	13
2 Using EVMS to Manage Devices	15
2.1 Configuring the System Device at Install to Use EVMS	15
2.1.1 Before the Install	15
2.1.2 During the Server Install	17
2.1.3 After the Server Install	20
2.2 Configuring an Existing System Device to Use EVMS	21
2.2.1 Disable the boot.lvm and boot.md Services	22
2.2.2 Enable the boot.evms Service	22
2.2.3 Edit the /etc/fstab File	23
2.2.4 Edit the Boot Loader File	24
2.2.5 Force the RAM Disk to Recognize the Root Partition	25
2.2.6 Reboot the Server	26
2.2.7 Verify that EVMS Manages the Boot, Swap, and Root Partitions	26
2.3 Configuring LVM Devices After Install to Use EVMS	26
2.4 Using EVMS with iSCSI Volumes	27
2.5 Using the ELILO Loader Files (IA-64)	27
2.6 Starting EVMS	28
2.7 Starting the EVMS Management Tools	28
3 Mounting EVMS File System Devices by UUIDs	29
3.1 Naming Devices with udev	29
3.2 Understanding UUIDs	29
3.2.1 Using UUIDs to Assemble or Activate File System Devices	30
3.2.2 Finding the UUID for a File System Device	30
3.3 Using UUIDs in the Boot Loader and /etc/fstab File (x86)	30
3.4 Using UUIDs in the Boot Loader and /etc/fstab File (IA64)	31
4 Managing Devices	33
4.1 Understanding Disk Segmentation	33
4.1.1 Segment Managers	33
4.1.2 Disk Segments	34
4.2 Initializing Disks	34
4.2.1 Before You Begin	34
4.2.2 Guidelines	35
4.2.3 Adding a Segment Manager	35
4.3 Removing the Segment Manager from a Device	36
4.4 Creating Disk Segments (or Partitions)	36

4.5	Configuring Mount Options for Devices	37
4.6	What's Next	39
5	Managing Multipath I/O for Devices	41
5.1	Understanding Multipathing	41
5.1.1	What Is Multipathing?	41
5.1.2	Benefits of Multipathing	42
5.1.3	Guidelines for Multipathing	42
5.1.4	Device Mapper	42
5.1.5	Device Mapper Multipath I/O Module	43
5.1.6	Multipath Tools	44
5.2	Before You Begin	44
5.2.1	Preparing SAN Devices for Multipathing	44
5.2.2	Partitioning Devices that Have Multiple Paths	45
5.2.3	Configuring mdadm.conf and lvm.conf to Scan Devices by UUID	45
5.3	Adding multipathd to the Boot Sequence	46
5.3.1	YaST	46
5.3.2	Command Line	46
5.4	Starting Multipath I/O Services	46
5.5	Configuring Time-Out Settings for the HBA	46
5.6	Configuring Multipath I/O for the Root Device	47
5.7	Scanning for New Devices without Rebooting	47
5.8	Configuring Multipathing for an Existing Software RAID	48
5.9	Configuring User-Friendly Names in the /etc/multipath.conf File	49
5.10	Managing I/O in Error Situations	50
5.11	Resolving Stalled I/O	50
5.12	Additional Information	51
5.13	What's Next	51
6	Managing Software RAID with EVMS	53
6.1	Understanding Software RAID on Linux	53
6.1.1	What Is a Software RAID?	53
6.1.2	Overview of RAID Levels	54
6.1.3	Comparison of RAID Performance	55
6.1.4	Comparison of Disk Fault Tolerance	55
6.1.5	Configuration Options for RAID	56
6.1.6	Guidelines for Component Devices	56
6.1.7	RAID 5 Algorithms for Distributing Stripes and Parity	57
6.1.8	Multi-Disk Plug-In for EVMS	59
6.1.9	Device Mapper Plug-In for EVMS	59
6.2	Creating and Configuring a Software RAID	59
6.3	Expanding a RAID	63
6.3.1	Adding Mirrors to a RAID 1 Device	63
6.3.2	Adding Segments to a RAID 4 or 5	64
6.4	Adding or Removing a Spare Disk	64
6.4.1	Do I Need a Spare Disk?	64
6.4.2	Adding a Spare Disk When You Create the RAID	65
6.4.3	Adding a Spare Disk to an Existing RAID	65
6.4.4	Removing a Spare Disk from a RAID	65
6.5	Managing Disk Failure and RAID Recovery	65
6.5.1	Understanding the Disk Failure and RAID Recovery	65
6.5.2	Identifying the Failed Drive	66
6.5.3	Replacing a Failed Device with a Spare	67

6.5.4	Removing the Failed Disk	68
6.6	Monitoring Status for a RAID	68
6.6.1	Monitoring Status with EVMSGUI	68
6.6.2	Monitoring Status with /proc/mdstat	68
6.6.3	Monitoring Status with mdadm	69
6.6.4	Monitoring a Remirror or Reconstruction.	71
6.6.5	Configuring mdadm to Send an E-Mail Alert for RAID Events	71
6.7	Deleting a Software RAID and Its Data	73
7	Managing Software RAID6 and 10 with mdadm	75
7.1	Creating a RAID 6	75
7.1.1	Understanding RAID 6	75
7.1.2	Creating a RAID 6	76
7.2	Creating Nested RAID 10 Devices with mdadm	76
7.2.1	Understanding Nested RAID Devices	76
7.2.2	Creating Nested RAID 10 (1+0) with mdadm	77
7.2.3	Creating Nested RAID 10 (0+1) with mdadm	78
7.3	Creating a Complex RAID 10 with mdadm	79
7.3.1	Understanding the mdadm RAID10	80
7.3.2	Creating a RAID10 with mdadm	82
7.4	Creating a Degraded RAID Array	83
8	Installing and Managing DRBD Services	85
8.1	Understanding DRBD	85
8.1.1	Additional Information	85
8.2	Installing DRBD Services.	85
8.3	Configuring the DRBD Service	86
9	Troubleshooting EVMS Devices, RAID6, and Volumes	89
9.1	EVMS Volumes Might Not Appear When Using iSCSI	89
9.2	Device Nodes Are Not Automatically Re-Created on Restart	89
A	Documentation Updates	91
A.1	February 1, 2007 (Updates).	91
A.1.1	Managing Software RAID6 with mdadm	91
A.2	December 1, 2006 (Updates)	92
A.2.1	Overview of EVMS	92
A.2.2	Using EVMS to Manage Devices	92
A.2.3	Managing Multipathing for Devices and Software RAID6	92
A.2.4	Managing Software RAID6	93
A.2.5	Managing Software RAID6 with mdadm	93

About This Guide

This guide provides information about how to manage storage devices on a SUSE® Linux Enterprise Server 10 server with the Enterprise Volume Management System (EVMS) 2.5.5. Related storage administration issues are also covered as noted below.

- ♦ Chapter 1, “Overview of EVMS,” on page 11
- ♦ Chapter 2, “Using EVMS to Manage Devices,” on page 15
- ♦ Chapter 3, “Mounting EVMS File System Devices by UUIDs,” on page 29
- ♦ Chapter 4, “Managing Devices,” on page 33
- ♦ Chapter 5, “Managing Multipath I/O for Devices,” on page 41
- ♦ Chapter 6, “Managing Software RAIDs with EVMS,” on page 53
- ♦ Chapter 7, “Managing Software RAIDs 6 and 10 with mdadm,” on page 75
- ♦ Chapter 8, “Installing and Managing DRBD Services,” on page 85
- ♦ Chapter 9, “Troubleshooting EVMS Devices, RAIDs, and Volumes,” on page 89

Audience

This guide is intended for system administrators.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation, or go to www.novell.com/documentation/feedback.html and enter your comments there.

Documentation Updates

For the most recent version of the *SUSE Linux Enterprise Server 10 Storage Administration Guide for EVMS*, visit the [Novell Documentation Web site for SUSE Linux Enterprise Server 10 \(http://www.novell.com/documentation/sles10\)](http://www.novell.com/documentation/sles10).

Additional Documentation

For information about managing storage with the Linux Volume Manager (LVM), see the *SUSE Linux Enterprise Server 10 Installation and Administration Guide* (<http://www.novell.com/documentation/sles10>).

Documentation Conventions

In Novell documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

Overview of EVMS

1

The Enterprise Volume Management System (EVMS) 2.5.5 management tool for Linux* is an extensible storage management tool that integrates all aspects of volume management, such as disk partitioning, the Logical Volume Manager (LVM), the Multiple-Disk (MD) manager for software RAIDs, the Device Mapper (DM) for multipath I/O configuration, and file system operations.

- ♦ [Section 1.1, “Benefits of EVMS,” on page 11](#)
- ♦ [Section 1.2, “Plug-In Layers,” on page 11](#)
- ♦ [Section 1.3, “File Systems Support,” on page 12](#)
- ♦ [Section 1.4, “Terminology,” on page 12](#)
- ♦ [Section 1.5, “Location of Device Nodes for EVMS Storage Objects,” on page 13](#)

1.1 Benefits of EVMS

EVMS provides the following benefits:

- ♦ Is an open source volume manager
- ♦ Provides a plug-in framework for flexible extensibility and customization
- ♦ Allows plug-ins to extend functionality for new or evolving storage managers
- ♦ Supports foreign partition formats
- ♦ Is cluster-aware

1.2 Plug-In Layers

EVMS abstracts the storage objects in functional layers to make storage management more user-friendly. The following table describes the current EVMS plug-in layers for managing storage devices and file systems:

Table 1-1 EVMS Plug-In Layers

Storage Managers	Description	Plug-Ins
Device	Manages the physical and logical devices	Device Mapper (DM)
Segment	Manages the partitioning of physical and logical devices into smaller segments of free space. Segment managers can be stacked. For example, a cluster segment can contain other storage objects or volumes.	Uses Device Mapper (DM) Segment managers include DOS, GPT, System/390* (S/390), Cluster, BSD, Mac, and BBR For more information, see Section 4.1, “Understanding Disk Segmentation,” on page 33 .

Storage Managers	Description	Plug-Ins
Regions	Manages the combination of multiple storage objects	LVM/LVM2 for containers and region, MD for RAIDs, and DM for multipath I/O
EVMS Features	Manages EVMS features	Drive linking (linear concatenation), Bad Block Relocation (BBR), and Snapshot
File System Interface Modules (FSIM)	Manages the interface between the file system managers and the segment managers	For information, see Section 1.3, "File Systems Support," on page 12.
Cluster Manager Interface Modules	Manages the interface between the cluster manager and the file systems and devices	HeartBeat 2

1.3 File Systems Support

EVMS supports the following Linux file systems:

- ♦ ReiserFS (the default file system for SUSE® Linux Enterprise Server 10)
- ♦ EXT3
- ♦ XFS
- ♦ OCFS2
- ♦ JFS
- ♦ EXT2
- ♦ Swap
- ♦ NTFS (read only)
- ♦ FAT (read only)

For more information about file systems supported in SUSE Linux Enterprise Server 10, see the *SUSE Linux Enterprise Server 10 Installation and Administration Guide*. (<http://www.novell.com/documentation/sles10>).

1.4 Terminology

EVMS uses the following terminology in the EVMS user interface:

Table 1-2 EVMS Terms

Term	Description
Sector	The lowest level that can be addressed on a block device.
Disk	A physical disk or a logical device.
Segment	An ordered set of physically contiguous sectors on a single device. It is similar to traditional disk partitions.

Term	Description
Region	An ordered set of logically contiguous sectors that might or might not be physically contiguous. The underlying mapping can be to logical disks, disk segments, or other storage regions.
Feature (Feature Object, EVMS Feature, EVMS Object)	A logically contiguous address space created from one or more disks, segments, regions, or other feature objects through the use of an EVMS feature.
Storage Object	Any storage structure in EVMS that is capable of being a block device. Disks, segments, regions, and feature objects are all storage objects.
Container (Storage Container)	<p>A collection of devices that are managed as a single pool of storage.</p> <p>Private Storage Container: A storage container that is exclusively owned and accessed by only one server.</p> <p>Cluster Storage Container: A storage container managed by the Cluster Resource Manager. It is accessible to all nodes of a cluster. An administrator can configure the storage objects in the cluster container from any node in the cluster. Cluster containers can be private, shared, or deported.</p> <ul style="list-style-type: none"> ♦ Private: The cluster container is exclusively owned and accessed by only one particular node of a cluster at any given time. The ownership can be reassigned by fail-over policies or the administrator. ♦ Shared: The cluster container is concurrently owned and accessed by all nodes of a cluster. Shared containers are preferred for distributed databases, clustered file systems, and cluster-aware applications that can coordinate safe access to shared volumes. ♦ Deported: The cluster container is not owned or accessed by any node of the cluster.
Volume (Logical Volume)	<p>A mountable storage object. Logical volumes can be EVMS volumes or compatibility volumes.</p> <ul style="list-style-type: none"> ♦ EVMS Volume: Volumes that contain EVMS metadata and support all EVMS features. Device nodes for EVMS volumes are stored in the <code>/dev/evms</code> directory. For example: <code>/dev/evms/my_volume</code> ♦ Compatibility Volume: Volumes that are backwards-compatible to other volume managers. They do not contain EVMS metadata and cannot support EVMS features.

1.5 Location of Device Nodes for EVMS Storage Objects

EVMS creates a unified namespace for the logical volumes on your system in the `/dev/evms` directory. It detects the storage objects actually present on a system, and creates an appropriate device node for each one, such as those shown in the following table.

Table 1-3 *Device Node Location*

Storage Object	Standard Location the Device Node	EVMS Location of the Device Node
A disk segment of disk	<code>/dev/sda5</code>	<code>/dev/evms/sda5</code>

Storage Object	Standard Location the Device Node	EVMS Location of the Device Node
A software RAID device	/dev/md1	/dev/evms/md/md1
An LVM volume	/dev/lvm_group/lvm_volume	/dev/evms/lvm/lvm_group/ lvm_volume

Using EVMS to Manage Devices

2

This section describes how to configure EVMS as the volume manager of your devices.

- ♦ [Section 2.1, “Configuring the System Device at Install to Use EVMS,” on page 15](#)
- ♦ [Section 2.2, “Configuring an Existing System Device to Use EVMS,” on page 21](#)
- ♦ [Section 2.3, “Configuring LVM Devices After Install to Use EVMS,” on page 26](#)
- ♦ [Section 2.4, “Using EVMS with iSCSI Volumes,” on page 27](#)
- ♦ [Section 2.5, “Using the ELILO Loader Files \(IA-64\),” on page 27](#)
- ♦ [Section 2.6, “Starting EVMS,” on page 28](#)
- ♦ [Section 2.7, “Starting the EVMS Management Tools,” on page 28](#)

2.1 Configuring the System Device at Install to Use EVMS

This section discusses how to configure the system device during the Linux install to use EVMS as the volume manager instead of the current default of Linux Volume Manager (LVM).

- ♦ [Section 2.1.1, “Before the Install,” on page 15](#)
- ♦ [Section 2.1.2, “During the Server Install,” on page 17](#)
- ♦ [Section 2.1.3, “After the Server Install,” on page 20](#)

2.1.1 Before the Install

- ♦ [“System Device” on page 15](#)
- ♦ [“Device Size Limits” on page 16](#)
- ♦ [“Data Loss Considerations for the System Device” on page 16](#)
- ♦ [“Storage Deployment Considerations for the System Device” on page 16](#)

System Device

For the purposes of this install documentation, a system device is any device that contains the Linux `/boot`, `swap`, or `root (/)` partitions for your Linux computer.

The install instructions assume the following:

- ♦ All three system partitions are on the same physical disk.

If you use different disks for any of the system partitions, make sure to modify the install instructions for your deployment scenario so that all of the system partitions are managed by EVMS.

- ♦ You must configure the boot partition within the BIOS-addressable space (such as 2 GB for x86 or 8 GB for x86-64) of the first disk recognized by the system.

If this restriction is not required for your hardware, you can modify the location of the `/boot` partition as desired.

- ♦ Your system uses the Grub or LILO boot loaders.

If you have an IA64 system, you must modify these install instructions to use the ELILO boot loader (`/boot/efi/elilo.conf`) instead.

WARNING: Whenever you manually alter the kernel or `initrd` on your system, make sure to run `/sbin/elilo` before shutting down the computer. If you leave out this step, your system might not be bootable.

Device Size Limits

Version 2.3 and later of `mdadm` supports component devices up to 4 TB in size each. Earlier versions support component devices up to 2 TB in size.

IMPORTANT: If you have a local disk, external disk arrays, or SAN devices that are larger than the supported device size, use a third-party disk partitioner to carve the devices into smaller logical devices.

You can combine up to 28 component devices to create the RAID array. The `md` RAID device you create can be up to the maximum device size supported by the file system you plan to use. For information about file system limits for SUSE® Linux Enterprise Server 10, see “Large File System Support” in the *SUSE Linux Enterprise Server 10 Installation and Administration Guide*. (<http://www.novell.com/documentation/sles10>).

Data Loss Considerations for the System Device

This install requires that you delete the default *Partitioning* settings created by the install, and create new partitions to use EVMS instead. This destroys all data on the disk.

WARNING: To avoid data loss, it is best to use the EVMS install option only on a new device.

If you have data volumes on the system device, take one or more of the following precautionary measures:

- ♦ Move the data volumes from the system device to another device.
- ♦ If you cannot move the volumes, make a backup copy of the data, so you can restore the data volumes later from a backup copy.

Storage Deployment Considerations for the System Device

By default, the YaST install for SUSE Linux Enterprise Server uses the Linux Volume Manager to manage the system device. The install procedures in this section describe how to install SUSE Linux Enterprise Server with EVMS as the volume manager of the system device. The instructions assume the following:

- ♦ You want to use EVMS to manage the system device.
- ♦ Only the system device is to be configured during the install.
- ♦ Other devices on the system are not configured during the install, or are attached to the server post-install. These additional devices are configured only after the system is operating and performing as expected.

2.1.2 During the Server Install

WARNING: The following install destroys all data on the system device.

To install Linux with EVMS as the volume manager for your boot and system partitions, you must modify the Partitioning configuration in the *Installation Settings*.

- 1** Begin the install, according to the instructions provided in the “Deployment” section of the *SUSE Linux Enterprise 10 Administration Guide*.
- 2** When the install process reaches the *Installations Settings* screen, delete the recommended partitions and the partition table on the system disk so that the device can be marked to use EVMS as the volume manager instead of LVM.
 - 2a** In the list of *Installation Settings*, select *Partitioning*.
 - 2b** In the *Partitioning* menu, select *Create Custom Partition Setup*, then click *Next*.
 - 2c** Select *Custom Partition - for Experts*, then click *Next* to open the Expert Partitioner options.
 - 2d** Select *Expert > Delete Partition Table and Disk Label*, then click *Yes* twice to continue through the Warning advisories.

This deletes the recommended partitions and the partition table on the system disk.

- 3** Create a primary partition on the system disk to use as the boot partition.

During the install, the `/boot` partition must remain under LVM control so that the install completes successfully. You do not modify the volume manager of the `/boot` partition until after the install is complete.

- 3a** Click *Create*.
- 3b** From the list of devices, select the device you want to use for the boot partition, such as `/dev/sda`, then click *OK*.

If you have a single system disk, only one device is available and you are not prompted for the device.
- 3c** Select *Primary Partition*, then click *OK*.
- 3d** Select *Format*, then select the native Linux file system you want to use, such as Reiser.
- 3e** In *Size (End Value)* field, specify 200 MB or larger.

For example, to set the size at 300 MB, type `300M`.
- 3f** Set the mount point to `/boot`.
- 3g** Click *OK*.

The partition appears as a logical device in the devices list, such as `/dev/sda1`.

- 4** Create a second primary partition on the system disk where you later create your swap and system volumes:

- 4a** Click *Create*.
- 4b** From the list of devices, select the device you want to use for the second primary partition, such as `/dev/sda`, then click *OK*.

If you have a single system disk, only one device is available and you are not prompted for the device.
- 4c** Select *Primary Partition*, then click *OK*.

- 4d** Select *Do Not Format*, then select *Linux LVM (0x8E)* from the list of file system IDs.
- 4e** In *Size*, set the cylinder *End Value* to 5 GB or larger, depending on the combined partition size you need to contain your system and swap volumes. You are creating a primary partition that becomes the EVMS container for these two volumes that you create later in this procedure.

IMPORTANT: Do not make the system partition larger than necessary. The remaining space on the system disk can be used to create data volumes that are managed by EVMS.

To determine the size you need, consider the following recommendations:

- ♦ If you intend to create data volumes on the same physical disk, you must leave unpartitioned space available.
- ♦ Set aside 128 MB or larger for the swap partition.

Swap management is different for Linux kernel 2.4.10 and later. How much swap to add depends on the RAM size, the tasks that are planned for the system, and whether you want to make more virtual memory available than the RAM provides.

Some swap (at least 128 MB) is good in order to minimize the risk of losing data when active processes run out of RAM space. Swap is not required for systems with more than 1 GB of RAM. You must have at least 1 GB of virtual memory (RAM plus swap) during the install, but if the swap is more than 2 GB, you might not be able to install on some machines.

- 4f** Click *OK*.

The partition appears as a logical device in the devices list, such as `/dev/sda2`.

- 5** Modify the volume management type from LVM to EVMS for the second primary partition you created in [Step 4](#) as follows:

- 5a** At the bottom of the page, click *EVMS*.

- 5b** In the EVMS Configuration dialog box, click *Create Container*, then select the LVM partition created in [Step 4](#).

- 5c** In the Create EVMS Container dialog box, specify the container name (such as *system*), then click *Add Volume* to create the `lvm2/system` container, where *system* is the container name.

- 5d** Click *OK*.

The EVMS Configuration dialog box now displays the `lvm2/system` container you just created, its size, and free space.

- 6** In the EVMS Configuration dialog box, create the *swap* volume in the `lvm2/system` container as follows:

- 6a** From the *EVMS Container* drop-down menu, select `lvm2/system`, then click *Add*.

- 6b** In the Create Logical Volume dialog box, select *Format*, then select *Swap* from the *File System* drop-down menu.

- 6c** Specify *swap* as the volume name.

- 6d** Specify the size of the *swap* volume, such as 1GB.

The *swap* volume should be at least 128 MB or larger. For more information, see [Step 4e on page 18](#).

- 6e** Specify the mount point as *swap*.

6f Click *OK*.

The swap volume is now listed as a volume in the `lvm2/system` container.

7 In the EVMS Configuration dialog box, create the root (`/`) volume in the `lvm2/system` container as follows:

7a From the *EVMS Container* drop-down menu, select `lvm2/system`, then click *Add*.

7b In the Create Logical Volume dialog box, select *Format*, then select the file system to use from the *File System* drop-down menu, such as *Reiser*.

7c In the *Volume Name* field, specify a volume name, such as `sys_lx`.

7d Specify the *Size* of the system volume as the remaining space available in the `lvm2/system` partition by clicking *Max*.

7e Specify the mount point as `/` (root volume).

7f Click *OK*.

The root (`/`) volume is now listed as a volume in the `lvm2/system` container.

8 Click *Next* to return to the list of devices.

Below is an example of the physical and logical devices that might be configured on your system. Your setup depends on the number of devices in the server and the sizes you choose for your partitions.

Device	Size	F	Type	Mount	Start	End	Used By
<code>/dev/sda</code>	149.0 GB		6Y160p0		0	19456	
<code>/dev/sda1</code>	305.9 MB	F	Linux Native (Reiser)	<code>/boot</code>	0	38	
<code>/dev/sda2</code>	20.0 GB		Linux LVM		39	2649	EVMS lvm2/ system
<code>/dev/sdb</code>	111.8 GB		SP1203N		0	14595	
<code>/dev/evms/lvm2/system/sys_lx</code>	10.0 GB	F	EVMS	<code>/</code>	-	-	
<code>/dev/evms/lvm2/system/swap</code>	1.0 GB	F	EVMS	<code>swap</code>	-	-	

9 Click *Next* to return to the Installation Settings page.

You can dismiss the message warning that you should not mix EVMS and non-EVMS partitions on the same device.

10 Continue with the installation.

IMPORTANT: After the install is complete, make sure to perform the mandatory post-install configuration of the related system settings to ensure that the system device functions properly under EVMS. Otherwise, the system fails to boot properly.

For information, see [“After the Server Install” on page 20](#).

2.1.3 After the Server Install

After the SUSE Linux Enterprise Server 10 install is complete, you must perform the following tasks to ensure that the system device functions properly under EVMS:

- ♦ “Edit the `/etc/fstab` File” on page 20
- ♦ “Disable the `boot.lvm` and `boot.md` Services” on page 21
- ♦ “Enable the `boot.evms` Service” on page 21
- ♦ “Reboot the Server” on page 21
- ♦ “Verify the System Services” on page 21

Edit the `/etc/fstab` File

When you boot the system, the kernel reads the `/etc/fstab` file to identify which file systems should be mounted and then mounts them. This file contains a table of file system information about the root (`/`), `/boot`, and `swap` partitions plus other partitions and file systems you want to mount.

The `/boot` partition is separate from the EVMS container where you placed the root (`/`) and `swap` partitions and is not managed by EVMS at this time. However, in the following steps, you disable `boot.lvm` and `boot.md`, then enable `boot.evms`. In effect, this forces EVMS to scan all the partitions at boot time, including the `/boot` partition, and it activates `/boot` under the `/dev/evms` directory. Therefore, this makes `/boot` a partition that is discovered by EVMS at startup, and requires that the device be listed under `/dev/evms` in the `fstab` file so it can be found when booting with `boot.evms`.

After the install, you must edit the `/etc/fstab` file to modify the location of the `/boot` partition so it is under the `/dev/evms` directory.

IMPORTANT: When working in the `/etc/fstab` file, do not leave any stray characters or spaces in the file. This is a configuration file, and it is highly sensitive to such mistakes.

1 Open the `/etc/fstab` file in a text editor.

2 Locate the line that contains the `/boot` partition.

For example, if your `/boot` partition uses device `sda1` and the *Reiser* file system, look for a line similar to this:

```
/dev/sda1 /boot reiser defaults 1 1
```

3 In the *Device Name* column, modify the location of the `/boot` partition from `/dev` to `/dev/evms` so it can be managed by EVMS. Modify only the device name by adding `/evms` to the path:

```
/dev/evms/sda1 /boot reiser defaults 1 1
```

4 Save the file.

The changes do not take affect until the server is restarted. Do not reboot at this time.

5 Continue with “Disable the `boot.lvm` and `boot.md` Services” on page 21.

Disable the boot.lvm and boot.md Services

Disable the `boot.lvm` and `boot.md` services so they do not run at boot time (runlevel B). EVMS now handles the boot.

- 1 In YaST, click *System > System Services (Runlevel) > Expert Mode*.
- 2 Select *boot.lvm*.
- 3 Click *Set/Reset > Disable the Service*.
- 4 Select *boot.md*.
- 5 Click *Set/Reset > Disable the Service*.
- 6 Click *Finish*, then click *Yes*.

The changes do not take affect until the server is restarted. Do not reboot at this time.

- 7 Continue with [“Enable the boot.evms Service” on page 21](#).

Enable the boot.evms Service

The `boot.evms` service should be enabled automatically after the install, but you should verify that it is enabled.

- 1 In YaST, click *System > System Services (Runlevel) > Expert Mode*.
- 2 Select *boot.evms*.
- 3 Click *Set/Reset > Enable the Service*.

The *B runlevel* option is automatically selected.

- 4 Click *Finish*, then click *Yes*.

The changes do not take affect until the server is restarted. You reboot in the next task.

NOTE: Effective in SUSE Linux Enterprise 10, the `/dev` directory is on `tmpfs` and the device nodes are automatically re-created on boot. It is no longer necessary to modify the `/etc/init.d/boot.evms` script to delete the device nodes on system reboot as was required for previous versions of SUSE Linux.

- 5 Continue with [“Reboot the Server” on page 21](#).

Reboot the Server

- 1 Reboot the server to apply the post-install configuration settings.

Verify the System Services

After the post-install configuration is complete and you have rebooted the server, make sure the server is operating as expected.

2.2 Configuring an Existing System Device to Use EVMS

If you have already installed Linux with a different volume manager for the system device (that is, the devices where you installed the `/boot`, `swap`, or `root (/)` partitions), you can optionally configure the device for EVMS at any time after the install.

If you do not configure the device to use EVMS, you must manage the device and all of its volumes with its current volume manager (the default is LVM), and free space on the device cannot be used for volumes you want to create using EVMS. Beginning with the Linux 2.6 kernel, any given device cannot be managed by multiple volume managers. However, you can have different volume managers for different devices.

The following procedures assume that you installed Linux with three partitions on a single SCSI device named `sda`:

```
/dev/sda1 reiserfs /boot
/dev/sda2 swap      swap
/dev/sda3 reiserfs /
```

IMPORTANT: Make sure to modify the following procedures as necessary for your specific setup.

- ♦ [Section 2.2.1, “Disable the boot.lvm and boot.md Services,” on page 22](#)
- ♦ [Section 2.2.2, “Enable the boot.evms Service,” on page 22](#)
- ♦ [Section 2.2.3, “Edit the /etc/fstab File,” on page 23](#)
- ♦ [Section 2.2.4, “Edit the Boot Loader File,” on page 24](#)
- ♦ [Section 2.2.5, “Force the RAM Disk to Recognize the Root Partition,” on page 25](#)
- ♦ [Section 2.2.6, “Reboot the Server,” on page 26](#)
- ♦ [Section 2.2.7, “Verify that EVMS Manages the Boot, Swap, and Root Partitions,” on page 26](#)

2.2.1 Disable the boot.lvm and boot.md Services

You need to disable `boot.lvm` (handles devices for Linux Volume Manager) and `boot.md` (handles multiple devices in software RAID) so they do not run at boot time. In the future, you want `boot.evms` to run at boot time instead.

- 1 In YaST, click *System > Runlevel Editor > Expert Mode*.
- 2 Select *boot.lvm*.
- 3 Click *Set/Reset > Disable the Service*.
- 4 Select *boot.md*.
- 5 Click *Set/Reset > Disable the Service*.
- 6 Click *Finish*, then click *Yes*.

The changes do not take affect until the server is restarted. Do not reboot at this time.

- 7 Continue with [Section 2.2.2, “Enable the boot.evms Service,” on page 22](#).

2.2.2 Enable the boot.evms Service

You need to enable the `boot.evms` service so that it boots devices when you restart the server.

- 1 In YaST, click *System > Runlevel Editor > Expert Mode*.
- 2 Select *boot.evms*.
- 3 Click *Set/Reset > Enable the Service*.

The *B runlevel* option is automatically selected.

- 4 Click *Finish*, then click *Yes*.

The changes do not take affect until the server is restarted. Do not reboot at this time.

NOTE: Effective in SUSE Linux Enterprise 10, the `/dev` directory is on `tmpfs` and the device nodes are automatically re-created on boot. It is no longer necessary to modify the `/etc/init.d/boot.evms` script to delete the device nodes on system reboot as was required for previous versions of SUSE Linux.

- 5 Continue with “[Edit the /etc/fstab File](#)” on page 23.

2.2.3 Edit the /etc/fstab File

When you boot the system, the kernel reads the `/etc/fstab` file to identify which file systems should be mounted and then mounts them. This file contains a table of file system information about the `/boot`, `swap`, and `root (/)` partitions plus other partitions and file systems you want to mount.

You must edit the `/etc/fstab` file to modify the mount location of these three partitions so they are mounted under the `/dev/evms` directory. For example, change `/dev/sda1` to `/dev/evms/sda1`.

Although the `/boot` partition is not managed by EVMS, the `boot.evms` script forces EVMS to scan all the partitions at boot time, including the `/boot` partition, and it activates `/boot` under the `/dev/evms` directory. Therefore, this makes `/boot` a partition that is discovered by EVMS at startup, and requires that the device’s path be listed under `/dev/evms` in the `fstab` file so it can be found when booting with `boot.evms`.

Make sure to replace `sda1`, `sda2`, and `sda3` with the device names you used for your partitions.

IMPORTANT: When working in the `/etc/fstab` file, do not leave any stray characters or spaces in the file. This is a configuration file, and it is highly sensitive to such mistakes.

- 1 Open the `/etc/fstab` file in a text editor.

- 2 Locate the line that contains the `/boot` partition.

For example, if your `/boot` partition uses device `sda1` and the *Reiser* file system, look for a line similar to this:

```
/dev/sda1 /boot reiser defaults 1 1
```

- 3 In the *Device Name* column, modify the mount location of the `/boot` partition from `/dev` to `/dev/evms` so it can be managed by EVMS. Modify only the device name by adding `/evms` to the path:

```
/dev/evms/sda1 /boot reiser defaults 1 1
```

- 4 Repeat [Step 2](#) and [Step 3](#) to edit the Device Name entry in the lines for the `swap` and `root (/)` partitions.

For example, change `/dev/sda2` to `/dev/evms/sda2`, and change `/dev/sda3` to `/dev/evms/sda3`.

- 5 Save the file.

The changes do not take affect until the server is restarted. Do not reboot at this time.

- 6 Continue with [Section 2.2.4, “Edit the Boot Loader File,”](#) on page 24.

2.2.4 Edit the Boot Loader File

When you boot the system, the kernel reads the boot loader file for information about your system. For Grub, this is the `/boot/grub/menu.lst` file. For LILO, this is the `/etc/lilo.conf` file.

You must edit the boot loader file to modify the mount location of partitions so they are mounted under the `/dev/evms` directory. For example, change `/dev/sda1` to `/dev/evms/sda1`. Make sure to replace the path for all lines that contain device paths in the files. You can modify the boot loader file by editing fields in YaST, or use a text editor to modify the file directly.

IMPORTANT: When working in the boot loader file, do not leave any stray characters or spaces in the file. This is a configuration file, and it is highly sensitive to such mistakes.

Using YaST

To modify the boot loader file in the YaST Control Center:

- 1** Log in as the `root` user or equivalent.
- 2** In Yast, select *System > Boot Loader*.
- 3** Modify the boot loader image so that the root file system is mounted as `/dev/evms/` instead of `/dev/`.
 - 3a** Select the boot loader image file, then click *Edit*.
 - 3b** Edit the device path in the *Root Device* field.

For example, change the *Root Device* value from

```
/dev/sda2
```

to

```
/dev/evms/sda2
```

Replace `sda2` with the actual device on your machine.
 - 3c** Edit any device paths in the *Other Kernel Parameters* field.
 - 3d** Click *OK* to save the changes and return to the Boot Loader page.
- 4** Modify the failsafe image so that the failsafe root file system is mounted as `/dev/evms/` instead of `/dev/`.
 - 4a** Select the failsafe image file, then click *Edit*.
 - 4b** Edit the device path in the *Root Device* field.
 - 4c** Check the *Other Kernel Parameters* field and make changes if needed.
 - 4d** Click *OK* to save the change and return to the Boot Loader page.
- 5** Click *Finish*.
- 6** Continue with [Section 2.2.5, “Force the RAM Disk to Recognize the Root Partition,” on page 25](#).

Using a Text Editor

To edit the boot loader file in a text editor:

- 1** Log in as the `root` user or equivalent.

- 2 Open the boot loader file in a text editor.

For Grub, this is the `/boot/grub/menu.1st` file. For LILO, this is the `/etc/lilo.conf` file.

- 3 Locate the line that contains the `root=` parameter.

For example, if your root file system uses device `sda1`, look for a line similar to this:

```
kernel (sd0,0)/vmlinuz root=/dev/sda1 vga=0x31a splash=silent
showopts
```

- 4 Modify the mount location from `/dev` to `/dev/evms` so it can be managed by EVMS.

For example, after the change, the line looks like this:

```
kernel (sd0,0)/vmlinuz root=/dev/evms/sda1 vga=0x31a splash=silent
showopts
```

- 5 Repeat **Step 3** and **Step 4** to locate other lines in the file that need to be similarly modified.

- 6 Save the file.

The changes do not take affect until the server is restarted. Do not reboot at this time.

- 7 Continue with **Section 2.2.5**, “Force the RAM Disk to Recognize the Root Partition,” on **page 25**.

2.2.5 Force the RAM Disk to Recognize the Root Partition

The `mkinitrd(8)` command creates file system images for use as initial RAM disk (`initrd`) images. These RAM disk images are often used to preload the block device modules (SCSI or RAID) needed to access the root file system.

You might need to force the RAM to update its device node information so that it loads the root (`/`) partition from the `/dev/evms` path.

NOTE: Recent patches to `mkinitrd` might resolve the need to do this task. For the latest version of `mkinitrd`, see *Recommended Updates for `mkinitrd`* (<http://support.novell.com/techcenter/psdb/24c7dfbc3e0c183970b70c1c0b3a6d7d.html>) at the Novell Technical Support Center.

- 1 At a terminal console prompt, enter the EVMS Ncurses command as the `root` user or equivalent:

```
evmsn
```

- 2 Review the output to verify that EVMS shows only the `/boot` and `swap` partitions as active in EVMS.

You should see the following devices mounted (with your own partition names, of course) for these two partitions:

```
/dev/evms/sda1
```

```
/dev/evms/sda2
```

- 3 At a terminal console prompt, enter the following update the `initrd` image with the `/dev/evms` path information for the root (`/`) partition:

```
/sbin/mkinitrd -f evms
```

This does not take effect until you restart the server.

- 4 Continue with **Section 2.2.6**, “Reboot the Server,” on **page 26**.

2.2.6 Reboot the Server

- 1 Reboot the server to apply the post-install configuration settings.

When your system reboots, the kernel loads the `init-ramdisk`, which runs the EVMS tools to activate your volumes and mount your root file system. Then your boot scripts run the EVMS tools once more to make sure your `/dev/evms/` directory correctly reflects the current state of your volumes. Finally, the remaining EVMS volumes are mounted as specified in your `/etc/fstab` file. Everything else on your system should start up as you would normally expect.

- 2 Continue with [Section 2.2.7, “Verify that EVMS Manages the Boot, Swap, and Root Partitions,”](#) on page 26.

2.2.7 Verify that EVMS Manages the Boot, Swap, and Root Partitions

- 1 At a terminal prompt, enter the EVMS Nurses command as the `root` user or equivalent:
`evmsn`
- 2 Review the output to verify that EVMS shows the `/boot`, `swap`, and `root (/)` partitions as active in EVMS.

You should see the following devices mounted (with your own partition names, of course) under the `/dev/evms` directory:

```
/dev/evms/sda1
/dev/evms/sda2
/dev/evms/sda3
```

2.3 Configuring LVM Devices After Install to Use EVMS

Use the following procedure to configure data devices (not system devices) to be managed by EVMS. If you need to configure an existing system device for EVMS, see [Section 2.2, “Configuring an Existing System Device to Use EVMS,”](#) on page 21.

- 1 In a terminal console, run the EVMSGUI by entering the following as the `root` user or equivalent:
`evmsgui`
- 2 In the *Volumes* panel, review the names that EVMS reports as compatibility volumes, find the devices that represent the devices you want to manage using EVMS, then write down the names for future reference.

For example, `/dev/sdb1`.

- 3 In a text editor, edit the `/etc/fstab` file to use the EVMS volume names.

For example, change the following entry for an LVM2 volume from this

```
/dev/sdb1 / reiserfs defaults 1 2
```

to this

```
/dev/evms/lvm2/sdb1 / reiserfs defaults 1 2
```

IMPORTANT: Make sure not to leave any stray characters or spaces in the line.

With these changes, each time your system boots, your file system is mounted using EVMS as the volume manager.

4 Update the boot scripts as follows:

- ♦ The command `evms_activate` must be run from your boot scripts in order to activate your volumes so they can be mounted.
- ♦ If you run software-RAID (`boot.md`) or LVM (`boot.lvm`) boot files in your boot scripts, and if you are moving all devices to EVMS, remove or disable those commands.

5 If you have not already done so, enable the `boot.evms` service.

For information, see [“Enable the boot.evms Service” on page 21](#).

6 Reboot your system.

2.4 Using EVMS with iSCSI Volumes

If your EVMS devices, RAIDs, and volumes use storage devices from an iSCSI SAN, make sure that your system starts iSCSI before EVMS so that the SAN and its disks are available to EVMS on system startup. iSCSI must be started and running before any disks/volumes on the iSCSI SAN can be accessed. If EVMS starts before iSCSI, EVMS cannot see or access the devices in the iSCSI SAN to mount the storage objects they contain, so the EVMS devices, RAIDs, and volumes might not be visible or accessible.

If EVMS starts before iSCSI on your system so that your EVMS devices, RAIDs, and volumes are not visible or accessible, you must correct the order in which iSCSI and EVMS are started. Enter the `chkconfig` command at the Linux server console of every server that is part of your iSCSI SAN.

1 At a terminal console prompt, enter either

```
chkconfig evms on
```

or

```
chkconfig boot.evms on
```

This ensures that EVMS and iSCSI start in the proper order each time your servers reboot.

2.5 Using the ELILO Loader Files (IA-64)

On a SUSE Linux Enterprise Server boot device EFI System Partition, the full paths to the loader and configuration files are:

```
/boot/efi/SuSE/elilo.efi
```

```
/boot/efi/SuSE/elilo.conf
```

When configuring partitioning during the install on IA64 systems, set the file system type for the `/boot` partition to `vfat`, then choose *Fstab Options* and set the *Arbitrary* option value to `umask=077` to ensure that the partition is accessible only to administrators.

WARNING: Whenever you manually alter the kernel or `initrd` on your system, make sure to run `/sbin/elilo` before shutting down the computer. If you leave out this step, your system might not be bootable.

2.6 Starting EVMS

If EVMS does not start during the system boot, you must activate it manually.

- 1 Open a terminal console, then log in as the `root` user or equivalent.
- 2 At the terminal console prompt, enter
`evms_activate`

2.7 Starting the EVMS Management Tools

Use the following procedure to start the EVMS management tools.

IMPORTANT: When you are done, make sure to exit the EVMS UI tool. When it is running, the EVMS UI tool locks the EVMS engine, potentially blocking other EVMS actions from taking place.

- 1 Open a terminal console, then log in as the `root` user or equivalent.
- 2 Enter one of the following commands to open the desired EVMS UI:

Command	Description
<code>evmsgui</code>	Starts the graphical interface for EVMS GUI. For information about features in this interface, see "EVMS GUI" (http://evms.sourceforge.net/user_guide/#GUI) in the <i>EVMS User Guide</i> at the EVMS project on SourceForge.net.
<code>evmsn</code>	Starts the text-mode interface for EVMS Ncurses. For information about features in this interface, see the "EVMS Ncurses Interface" (http://evms.sourceforge.net/user_guide/#NCURSES) in the <i>EVMS User Guide</i> at the EVMS project on SourceForge.net.
<code>evms</code>	Starts the EVMS command-line interpreter (CLI) interface. For information about command options, see "EVMS Command Line Interpreter" (http://evms.sourceforge.net/user_guide/#COMMANDLINE) in the <i>EVMS User Guide</i> at the EVMS project on SourceForge.net.

To stop `evmsgui` from running automatically on reboot:

- 1 Close `evmsgui`.
- 2 Do a clean shutdown (not a restart).
- 3 Start the server.

When the server comes back up, `evmsgui` is not automatically loaded on reboot.

Mounting EVMS File System Devices by UUIDs

3

This section discusses the optional use of UUIDs instead of device names to identify file system devices in the boot loader file and the `/etc/fstab` file.

- ♦ [Section 3.1, “Naming Devices with udev,” on page 29](#)
- ♦ [Section 3.2, “Understanding UUIDs,” on page 29](#)
- ♦ [Section 3.3, “Using UUIDs in the Boot Loader and `/etc/fstab` File \(x86\),” on page 30](#)
- ♦ [Section 3.4, “Using UUIDs in the Boot Loader and `/etc/fstab` File \(IA64\),” on page 31](#)

3.1 Naming Devices with udev

In the Linux 2.6 and later kernel, `udev` provides a userspace solution for the dynamic `/dev` directory, with persistent device naming. As part of the hotplug system, `udev` is executed if a device is added or removed from the system.

A list of rules is used to match against specific device attributes. The `udev` rules infrastructure (defined in the `/etc/udev/rules.d` directory) provides stable names for all disk devices, regardless of their order of recognition or the connection used for the device. The `udev` tools examine every appropriate block device that the kernel creates to apply naming rules based on certain buses, drive types, or file systems. For information about how to define your own rules for `udev`, see [Writing udev Rules \(http://reactivated.net/writing_udev_rules.html\)](http://reactivated.net/writing_udev_rules.html).

Along with the dynamic kernel-provided device node name, `udev` maintains classes of persistent symbolic links pointing to the device in the `/dev/disk` directory, which is further categorized by the `by-id`, `by-label`, `by-path`, and `by-uuid` subdirectories.

NOTE: Other programs besides `udev`, such as LVM or `md`, might also generate UUIDs, but they are not listed in `/dev/disk`.

For more information about `udev(8)`, see its man page. Enter the following at a terminal console prompt:

```
man 8 udev
```

3.2 Understanding UUIDs

A UUID (Universally Unique Identifier) is a 128-bit number for a file system that is unique on both the local system and across other systems. It is a randomly generated with system hardware information and time stamps as part of its seed. UUIDs are commonly used to uniquely tag devices.

- ♦ [Section 3.2.1, “Using UUIDs to Assemble or Activate File System Devices,” on page 30](#)
- ♦ [Section 3.2.2, “Finding the UUID for a File System Device,” on page 30](#)

3.2.1 Using UUIDs to Assemble or Activate File System Devices

The UUID is always unique to the partition and does not depend on the order in which it appears or where it is mounted. With certain SAN devices attached to the server, the system partitions are renamed and moved to be the last device. For example, if root (/) is assigned to `/dev/sda1` during the install, it might be assigned to `/dev/sdg1` after the SAN is connected. One way to avoid this problem is to use the UUID in the boot loader and `/etc/fstab` files for the boot device.

A UUID never changes, no matter where the device is mounted, so it can always be found at boot. In a boot loader file, you typically specify the location of the device (such as `/dev/sda1` or `/dev/evms/sda1`) to mount it at system boot. The boot loader can also mount devices by their UUIDs and administrator-specified volume labels. However, if you use a label and file location, you cannot change the label name when the partition is mounted.

You can use the UUID as criterion for assembling and activating software RAID devices. When a RAID is created, the md driver generates a UUID for the device, and stores the value in the md superblock.

3.2.2 Finding the UUID for a File System Device

You can find the UUID for any block device in the `/dev/disk/by-uuid` directory. For example, a UUID looks like this:

```
e014e482-1c2d-4d09-84ec-61b3aefde77a
```

3.3 Using UUIDs in the Boot Loader and `/etc/fstab` File (x86)

After the install, you can optionally use the following procedure to configure the UUID for the system device in the boot loader and `/etc/fstab` files for your x86 system.

- 1** Install the SUSE® Linux Enterprise Server for x86 with no SAN devices connected.
- 2** After the install, boot the system.
- 3** Open a terminal console as the `root` user or equivalent.
- 4** Navigate to the `/dev/disk/by-uuid` directory to find the UUID for the device where you installed `/boot`, `/root`, and `swap`.
 - 4a** At the terminal console prompt, enter

```
cd /dev/disk/by-uuid
```
 - 4b** List all partitions by entering

```
ll
```
 - 4c** Find the UUID, such as

```
e014e482-1c2d-4d09-84ec-61b3aefde77a -> /dev/sda1
```
- 5** Edit `/boot/grub/menu.lst` file, using the Boot Loader option in YaST2 or using a text editor.

For example, change

```
kernel /boot/vmlinuz root=/dev/sda1
```

to

```
kernel /boot/vmlinuz root=/dev/disk/by-uuid/e014e482-1c2d-4d09-84ec-61b3aefde77a
```

IMPORTANT: Make a copy of the original boot entry, then modify the copy. If you make a mistake, you can boot the server without the SAN connected, and fix the error.

If you use the Boot Loader option in YaST, there is a defect where it adds some duplicate lines to the boot loader file when you change a value. Use an editor to remove the following duplicate lines:

```
color white/blue black/light-gray
default 0
timeout 8
gfxmenu (sd0,1)/boot/message
```

When you use YaST to change the way that the root (/) device is mounted (such as by UUID or by label), the boot loader configuration needs to be saved again to make the change effective for the boot loader.

- 6** As the `root` user or equivalent, do one of the following to place the UUID in the `/etc/fstab` file:

- ♦ Open YaST to *System > Partitioner*, select the device of interest, then modify *Fstab Options*.
- ♦ Edit the `/etc/fstab` file to modify the system device from the location to the UUID.

For example, if the root (/) volume has a device path of `/dev/sda1` and its UUID is `e014e482-1c2d-4d09-84ec-61b3aefde77a`, change line entry from

```
/dev/sda1 / reiserfs acl,user_xattr 1 1
```

to

```
UUID=e014e482-1c2d-4d09-84ec-61b3aefde77a / reiserfs
acl,user_xattr 1 1
```

IMPORTANT: Make sure to make a backup copy of the `fstab` file before you begin, and do not leave stray characters or spaces in the file.

3.4 Using UUIDs in the Boot Loader and `/etc/fstab` File (IA64)

After the install, use the following procedure to configure the UUID for the system device in the boot loader and `/etc/fstab` files for your IA64 system. IA64 uses the EFI BIOS. Its file system configuration file is `/boot/efi/SuSE/elilo.conf` instead of `/etc/fstab`.

- 1** Install the SUSE Linux Enterprise Server for IA64 with no SAN devices connected.
- 2** After the install, boot the system.
- 3** Open a terminal console as the `root` user or equivalent.
- 4** Navigate to the `/dev/disk/by-uuid` directory to find the UUID for the device where you installed `/boot`, `/root`, and `swap`.
 - 4a** At the terminal console prompt, enter

```
cd /dev/disk/by-uuid
```

4b List all partitions by entering

```
ll
```

4c Find the UUID, such as

```
e014e482-1c2d-4d09-84ec-61b3aefde77a -> /dev/sda1
```

5 Edit the boot loader file, using the Boot Loader option in YaST2.

For example, change

```
root=/dev/sda1
```

to

```
root=/dev/disk/by-uuid/e014e482-1c2d-4d09-84ec-61b3aefde77a
```

6 Edit the `/boot/efi/SuSE/elilo.conf` file to modify the system device from the location to the UUID.

For example, change

```
/dev/sda1    /    reiserfs    acl,user_xattr          1 1
```

to

```
UUID=e014e482-1c2d-4d09-84ec-61b3aefde77a    /    reiserfs  
acl,user_xattr          1 1
```

IMPORTANT: Make sure to make a backup copy of the `/boot/efi/SuSE/elilo.conf` file before you begin, and do not leave stray characters or spaces in the file.

Managing Devices

4

This section discusses how to initialize a disk by adding a segment management container to manage the partitions that you later add to the disk.

- ♦ [Section 4.1, “Understanding Disk Segmentation,” on page 33](#)
- ♦ [Section 4.2, “Initializing Disks,” on page 34](#)
- ♦ [Section 4.3, “Removing the Segment Manager from a Device,” on page 36](#)
- ♦ [Section 4.4, “Creating Disk Segments \(or Partitions\),” on page 36](#)
- ♦ [Section 4.5, “Configuring Mount Options for Devices,” on page 37](#)
- ♦ [Section 4.6, “What’s Next,” on page 39](#)

4.1 Understanding Disk Segmentation

In EVMS, you initialize a disk by assigning a segment manager to it. The segment manager creates metadata for the disk and exposes its free space so you can subdivide it into one or multiple segments (also called partitions).

- ♦ [Section 4.1.1, “Segment Managers,” on page 33](#)
- ♦ [Section 4.1.2, “Disk Segments,” on page 34](#)

4.1.1 Segment Managers

The most commonly used segment manager is the DOS Segment Manager. The following table describes the segment managers available in EVMS.

Table 4-1 *EVMS Segment Managers*

Segment Manager	Description
DOS	The standard MS-DOS disk partitioning scheme. It is the most commonly used partitioning scheme for Linux, NetWare®, Windows*, OS/2*, BSD, SolarisX86, and UnixWare*.
GPT (Globally Unique Identifier (GUID) Partitioning Table)	<p>A partitioning scheme used for IA-64 platforms, as defined in the Intel* <i>Extensible Firmware Interface (EFI) Specification</i>. It is not compatible with DOS, Windows, or OS/2 systems.</p> <p>The GUID is also known as Universally Unique Identifier (UUID). The GPT combines time and space descriptors to create this unique 128-bit tag for the disk and its segments.</p>
S/390	A partitioning scheme used exclusively for System/390 mainframes.
Cluster	A partitioning scheme for high-availability clusters. It provides a GUID for the disk, creates an EVMS container for the shared cluster devices, and specifies a node ID for the node that owns the device and the cluster ID.
BSD	A partitioning scheme for BSD UNIX.

Segment Manager	Description
MAC	A partitioning scheme for Mac-OS partitions.

4.1.2 Disk Segments

After you initialize the disk by adding a segment manager, you see metadata and free space segments on the disk. You can then create one or multiple data segments in a disk segment.

Table 4-2 *Disk Segment Types*

Segment Type	Description
Metadata	A set of contiguous sectors that contain information needed by the segment manager.
Free Space	A set of contiguous sectors that are unallocated or not in use. Free space can be used to create a segment.
Data	A set of contiguous sectors that has been allocated from a disk. The segment might be in use for a volume or a software RAID.

4.2 Initializing Disks

You must initialize new disks and disks that you want to reformat. After the disk is initialized, you can subdivide, or carve, the device into one or more disk segments for your file systems.

- [Section 4.2.1, “Before You Begin,” on page 34](#)
- [Section 4.2.2, “Guidelines,” on page 35](#)
- [Section 4.2.3, “Adding a Segment Manager,” on page 35](#)

4.2.1 Before You Begin

If you use large disks or disk arrays, use the vendor’s tools to carve them into the sizes that are usable for the management tools you plan to use. For example, the `md` driver recognizes disks only up to 2 TB in size, so the limit also applies to the `md` plug-in for EVMS. Software RAID devices you create with EVMS can be larger than 2 TB, of course, because the `md` driver plug-in manages the disks underneath that storage structure.

When you boot the server, EVMS scans and recognizes all devices it manages. If you add a new device to the server or create a device using `mkfs`, EVMS automatically mounts it on reboot under `/dev/evms` as a compatibility volume, such as `/dev/evms/sdb`.

IMPORTANT: If you cannot find a new disk, device, or volume, look under `/dev/evms` in a file browser, or look for compatibility volumes in the Volumes Manager in the EVMS GUI (`evmsgui`).

4.2.2 Guidelines

Consider the following guidelines when initializing a disk:

- ♦ EVMS might allow you to create segments without first adding a segment manager for the disk, but it is best to explicitly add a segment manager to avoid problems later.

IMPORTANT: You must add a Cluster segment manager if you plan to use the devices for volumes that you want to share as cluster resources.

- ♦ When you initialize an existing disk that is already formatted, the process of adding a segment manager destroys all data on the disk. If you want to keep the data on the disk, make sure to back up the data before you begin this process.
- ♦ For existing disks on the system or disks that you move from another system, you must delete any existing volume management structures, and remove any segment managers. This removes the device's metadata and data, and destroys all data on the disk.

WARNING: Do not initialize the device that contains your current system disk or any device that contains the `/boot`, `swap`, or `root (/)` volumes.

- ♦ If a new disk does not show up in the list of *Available Objects*, look for it in the *Volumes* list to see if the disk shows up as a compatibility volume. For example, a new disk `sdb` would show up as `/dev/evms/sdb`. Delete it from the *Volumes* list to force the disk to show up in *Available Objects*, then create segments as desired.

4.2.3 Adding a Segment Manager

Use the following procedure to assign a segment manager to device for servers using x86, x64, and IA64 controllers. This option is not available for S390 platforms, so simply continue with configuring software RAIDs or file system partitions, as desired.

WARNING: Adding a segment manager initializes the disk, completely removing all the segments it contains. All the data stored on the device is lost.

- 1 If the disk has any existing volume management structures or an existing segment manager, remove them.
 - 1a Select *Actions > Delete > Volume* to view the *Volumes* list.
 - 1b Select any existing volume management structures on the device, then click *Delete*.
 - 1c Select *Actions > Remove > Segment Manager from Storage Object*.
 - 1d Select the type of Segment Manager in use, then click *Next*.
 - 1e Select the device, then click *Remove*.
- 2 If the disk is a new one that is listed as a compatibility volume in the *Volumes* list, delete it as a compatibility volume.
 - 2a Select *Actions > Delete > Volume* to view the *Volumes* list.
 - 2b Select the device, then click *Delete*.
- 3 Add the Segment Manager.
 - 3a In the list of *Availability Objects*, select the device, then click *Actions > Add > Segment Manager to Storage Object*.

- 3b** From the list, select one of the following types of segment manager, then click *Next*.
 - ♦ *DOS Segment Manager* (the most common choice)
 - ♦ *GPT Segment Manager* (for IA-64 platforms)
 - ♦ *Cluster Segment Manager* (available only if it is a viable option for the selected disk)
- 3c** Select the device from the list of *Plugin Acceptable Objects*, then click *Next*.
- 3d** If required, specify the disk type as Linux.
- 3e** Click *Add* to create the segment management container for the disk, then click OK to dismiss the confirmation message.

4.3 Removing the Segment Manager from a Device

- 1** If the disk has any existing volume management structures, remove them.
 - 1a** Select *Actions > Delete > Volume* to view the Volumes list.
 - 1b** Select any existing volume management structures on the device, then click *Delete*.
- 2** Select *Actions > Remove > Segment Manager from Storage Object*.
- 3** Select the type of Segment Manager in use, then click *Next*.
- 4** Select the device, then click *Remove*.

4.4 Creating Disk Segments (or Partitions)

- 1** In EVMS, select *Actions > Create > Segment* to see a list of segment managers.
 - 2** From the list, select the segment manager for the device you want to manage, then click *Next*.
 - ♦ *DOS Segment Manager* (the most common choice)
 - ♦ *GPT Segment Manager* (for IA-64 platforms)
 - ♦ *Cluster Segment Manager* (available only if it is a viable option for the selected disk)
- For information about these and other segment managers available, see [“Segment Managers” on page 33](#).

- 3** Select the storage object that you want to segment, then click *Next*.
- 4** Complete the required configuration options for the segment, and modify default values as desired.
 - ♦ *Size (MB)*: Specify the amount of space (in MB) that you want to use. Use the arrows or type in a value. The interface corrects the value to the lower or upper size limit if you specify a size that is too small or that exceeds the amount of free space available.
 - ♦ *Offset (sectors)*: Specify the number of sectors to skip before beginning this partition if you want to leave free space in front of it.
 - ♦ *Partition Type*: From the drop-down list, select *Linux* (default), *Linux Swap*, *Linux LVM*, *NTFS*, *HPFS*, *FAT16*, or *Other Partition Type*.
 - ♦ *Partition Type ID*: This value changes automatically based on the *Partition Type* value, except for the *Other Partition Type* option. Then you must manually enter a value.
 - ♦ *Bootable*: Click *Yes* to make a primary partition active so that you can boot from it, or click *No* to make it unbootable. No is the only option if you are creating a logical partition.

- ♦ *Primary Partition*: Click *Yes* for a primary partition, or click *No* for a logical partition.

Required settings are denoted in the page by an asterisk (*). All required fields must be completed to make the *Create* button active.

- 5 Click *Create* to create the segment.
- 6 Verify that the new segment appears in the Segment list.

4.5 Configuring Mount Options for Devices

The following table describes the *Fstab Options* that are configurable in YaST. The values are written to the `/etc/fstab` file and are applied upon reboot.

Table 4-3 *Fstab Options in YaST*

Fstab Option	Description
Mount by	<ul style="list-style-type: none"> ♦ Device name (K) (default, such as <code>/dev/sda2</code>) ♦ Volume label (L) Make sure to also specify a value in <i>Volume Label</i>. ♦ UUID (U) For information about why you might want to discover partitions and devices by UUID, see Section 3.2.1, “Using UUIDs to Assemble or Activate File System Devices,” on page 30. ♦ Device ID (I) ♦ Device Path (P)
Volume label	A useful name to help you easily identify the volume on the server. By default, this field is empty.
Mount read-only	<p>Select the check box to enable this option. It is deselected (disabled) by default.</p> <p>If this option is enabled, files and directories cannot be modified or saved on the volume.</p>
No access time	<p>Select the check box to enable this option. It is deselected (disabled) by default.</p> <p>By default, the Linux <code>open(2)</code> command updates the access time whenever a file is opened. The <i>No Access Time</i> option disables the updating of access time, so that reading a file does not update its access time. Enabling the <i>No Access Time</i> option allows you to back up a volume without modifying the access times of its files.</p>
Mountable by user	<p>Select the check box to enable this option. It is deselected (disabled) by default.</p> <p>If this option is enabled, the volume can be mounted by any user; <code>root</code> privileges are not required.</p>
Do Not Mount at System Start-up	<p>Select the check box to enable this option. It is deselected (disabled) by default.</p> <p>The system volumes such as <code>/boot</code>, <code>swap</code>, and <code>root (/)</code> should all be mounted at system start. For other volumes, enable this option for a volume if you want to mount it manually later using the <code>mount</code> command at a terminal console prompt.</p>

Fstab Option	Description
Data journaling mode	<p>For journaling file systems, select the preferred journaling mode: ordered (default), journal, or writeback.</p> <ul style="list-style-type: none"> ♦ Ordered: Writes data to the file system, then enters the metadata in the journal. ♦ Journal: Writes data twice; once to the journal, then to the file system. ♦ Writeback: Writes data to the file system and writes metadata in the journal, but the writes are performed in any order.
Access Control Lists (ACL)	Select this option to enable access control lists on the file system. It is enabled by default.
Extended user attributes	Select this option to enable extended user attributes on the file system. It is enabled by default.
Arbitrary option value	Specify any mount option that is legal for the Mount Options column for a device entry in the <code>/etc/fstab</code> file. Use a comma with no spaces to separate multiple options.

You can modify these values for each entry by editing the `/etc/fstab` file, or use the following procedure to modify the mount options for a volume in the `/etc/fstab` file from YaST.

- 1 Open YaST, then click *System > Partitioning*.
- 2 Select the device you want to modify, then click *Fstab Options*.

Mount in /etc/fstab By: Normally, a file system to mount is identified in `/etc/fstab` by the device name. This identification can be changed so the file system to mount is found by searching for a UUID or a volume label. Not all file systems can be mounted by UUID or a volume label. If an option is disabled, it is not possible.

Volume label: The name entered in this field is used as the volume label. This normally only makes sense when you activate the option for mounting by volume label. In the volume label is not possible to use character `/` and space.

Mount Read-Only: No writable access to the file system is possible. Default is false.

No access time: Access times are not updated when a file is read. Default is false.

Mountable by User: The file system may be mounted by an ordinary user. Default is false.

Not Mounted at System Start-up: The file system is not automatically mounted when the system starts. An entry in `/etc/fstab` is created and the file system is mounted with the appropriate options when the command `mount <mount point>` (`<mount point>` is the directory to which the

Fstab options:

Mount in /etc/fstab by

☐ Device name
 ☐ Device ID
 ☐ Volume label
 ☐ Device Path
 ☒ UUID

Volume Label

Root

☐ Mount read-only
 ☐ No access time
 ☐ Mountable by user
 ☐ Do Not Mount at System Start-up

Data Journaling Mode

ordered

☒ Access Control Lists (ACL)
 ☒ Extended User Attributes

Arbitrary option value

OK Cancel

3 Modify the settings as desired, then click *OK* to accept your changes.

4.6 What's Next

If multiple paths exist between your host bus adapters (HBAs) and the storage devices, configure multipathing for the devices before creating software RAIDs or file system volumes on the devices. For information, see [Chapter 5, “Managing Multipath I/O for Devices,” on page 41](#).

If you want to configure software RAIDs, do it before you create file systems on the devices. For information, see [Chapter 6, “Managing Software RAIDs with EVMS,” on page 53](#).

Managing Multipath I/O for Devices

5

This section discusses how to configure multiple paths between the servers and storage devices for automatic failover and optional load balancing.

- ♦ [Section 5.1, “Understanding Multipathing,” on page 41](#)
- ♦ [Section 5.2, “Before You Begin,” on page 44](#)
- ♦ [Section 5.3, “Adding multipathd to the Boot Sequence,” on page 46](#)
- ♦ [Section 5.4, “Starting Multipath I/O Services,” on page 46](#)
- ♦ [Section 5.5, “Configuring Time-Out Settings for the HBA,” on page 46](#)
- ♦ [Section 5.6, “Configuring Multipath I/O for the Root Device,” on page 47](#)
- ♦ [Section 5.7, “Scanning for New Devices without Rebooting,” on page 47](#)
- ♦ [Section 5.8, “Configuring Multipathing for an Existing Software RAID,” on page 48](#)
- ♦ [Section 5.9, “Configuring User-Friendly Names in the /etc/multipath.conf File,” on page 49](#)
- ♦ [Section 5.10, “Managing I/O in Error Situations,” on page 50](#)
- ♦ [Section 5.11, “Resolving Stalled I/O,” on page 50](#)
- ♦ [Section 5.12, “Additional Information,” on page 51](#)
- ♦ [Section 5.13, “What’s Next,” on page 51](#)

5.1 Understanding Multipathing

- ♦ [Section 5.1.1, “What Is Multipathing?,” on page 41](#)
- ♦ [Section 5.1.2, “Benefits of Multipathing,” on page 42](#)
- ♦ [Section 5.1.3, “Guidelines for Multipathing,” on page 42](#)
- ♦ [Section 5.1.4, “Device Mapper,” on page 42](#)
- ♦ [Section 5.1.5, “Device Mapper Multipath I/O Module,” on page 43](#)
- ♦ [Section 5.1.6, “Multipath Tools,” on page 44](#)

5.1.1 What Is Multipathing?

Multipathing is the ability of a server to communicate with the same physical or logical storage device across multiple physical paths between host bus adapters in the server and the storage controllers for the device, typically in Fibre Channel (FC) or iSCSI SAN environments. You can also achieve multiple connections with direct attached storage for SCSI connections. Multipathing provides connection fault tolerance and can optionally provide load balancing across the available connections.

Multipathing is managed at the device level. It is not possible to manage connections to Linux partitions contained in the device. You can combine multiple devices to create a software RAID or volumes as logical devices, but the physical device itself is managed at a lower level.

IMPORTANT: If you plan to use multipathing with software RAIDs, you should first configure multipathing for the devices to create the multipath storage object for each device, then configure the RAID with them.

5.1.2 Benefits of Multipathing

Multipathing can help provide fault tolerance for the connection between the server and its storage devices, typically in a SAN configuration. When multipathing is configured and running, it automatically isolates and identifies device connection failures, and reroutes I/O to alternate connections. A previously failed path is automatically reinstated when it becomes healthy again.

Typical connection problems involve faulty adapters, cables, or controllers. When you configure multipath I/O for a device, the multipath driver monitors the active connection between devices. When it detects I/O errors, the multipath driver fails over to a designated secondary path. When the primary path recovers, control is automatically returned to the primary connection.

5.1.3 Guidelines for Multipathing

Multipathing is available only under the following conditions:

- ♦ Multiple physical paths must exist between host bus adapters in the server and host bus controllers for the storage device.
- ♦ Device partitioning (disk carving) and hardware RAID configuration should be completed prior to configuring multipathing. If you change the partitioning in the running system, Device Mapper Multipath I/O (DM-MPIO) does not automatically detect and reflect these changes. It must be reinitialized, which usually requires a reboot.
- ♦ For software RAID devices, multipathing should be configured prior to creating the software RAID devices because multipathing runs underneath the software RAID.
- ♦ In the initial release of SUSE Linux Enterprise Server 10, DM-MPIO is not available for the boot partition, because the boot loader cannot handle multipath I/O. Therefore, we recommend you set up a separate boot (`/boot`) partition when using multipathing. This has been resolved in the latest package updates.
- ♦ The storage subsystem you use on the multipathed device must support multipathing. Most storage subsystems should work; however, they might require an appropriate entry in the `DEVICE` variable in the `/etc/multipath.conf` file.

For a list of supported storage subsystems that allows multiple paths to be detected automatically, see “10.1 Supported Hardware” in the *SUSE Linux Enterprise Server 10 Administration Guide* (http://www.novell.com/documentation/sles10/sles_admin/data/sec_mpio_supported_hw.html).

- ♦ When configuring devices for multipathing, use the device names in the `/dev/disk/by-id` directory instead of the default device names (such as `/dev/sd*`), because the `/dev/disk/by-id` names persist over reboots.

5.1.4 Device Mapper

The default settings for `mdadm.conf` (and `lvm.conf`) do not work properly with multipathed devices. By default, both `md` and `LVM2` scan physical devices only and ignore any symbolic links or device-mapper devices.

This does not work for multipathed devices as there we have to omit all physical devices and scan devices in `/dev/disk/by-id` only as these are the correct multipathed devices.

If a previous MD installation exists, modify `mdadm.conf` to handle the devices correctly by ID instead of by device node path. For instructions, see [Section 5.2.3, “Configuring mdadm.conf and lvm.conf to Scan Devices by UUID,”](#) on page 45.

5.1.5 Device Mapper Multipath I/O Module

The Device Mapper Multipath I/O (DM-MPIO) module provides the multipathing capability for Linux. Multipath protects against failures in the paths to the device, and not failures in the device itself. If one of the paths is lost (for example, a network adapter breaks or a fiber-optic cable is removed), I/O will be redirected to the remaining paths. If an active path fails, the DM continues to balance traffic across the healthy paths. If all active paths fail, inactive secondary paths must be waked up, so failover occurs with a delay of approximately 30 seconds.

Table 5-1 *Multipath I/O Features*

Features	Description
Active/passive	If the storage array has multiple controllers, and only one controller is active at a time, then only the paths from the host to the active storage controller are active. Connections to the second and subsequent controllers are passive.
Active/active	If the storage array has multiple controllers that are concurrently active, all connections from the host to the controllers are active and treated equally in a load-balanced setup.
Load balancing	The Device Mapper driver automatically load balances traffic across all active paths.
Controller failover	When the active controller fails over to the passive, or standby, controller, the Device Mapper driver automatically activates the paths between the host and the standby, making them the primary paths. When the failed primary controller is reactivated as primary, the Device Mapper driver automatically activates the previously-downed paths, too.
Boot/Root device support	Multipathing is supported for the root (<code>/</code>) device in SUSE Linux Enterprise Server 10 and later.

Device Mapper detects every path for a multipathed device as a separate SCSI device. The SCSI device names take the form `/dev/sdN`, where *N* is an autogenerated letter for the device, beginning with a and issued sequentially as the devices are created, such as `/dev/sda`, `/dev/sdb`, and so on. If the number of devices exceeds 26, the letters are duplicated such that the next device after `/dev/sdz` will be named `/dev/sdaa`, `/dev/sdab`, and so on.

If multiple paths are not automatically detected, you can configure them manually in the `/etc/multipath.conf` file.

5.1.6 Multipath Tools

The `multipath-tools` user-space package takes care of automatic path discovery and grouping. It automatically tests the path periodically, so that a previously failed path is automatically reinstated when it becomes healthy again. This minimizes the need for administrator attention in a production environment.

The tools are described in the following table:

Table 5-2 Tools Available in the `multipath-tools` Package

Tool	Description
<code>multipath</code>	Scans the system for multipathed devices and assembles them.
<code>multipathd</code>	Waits for maps events then executes <code>multipath</code> .
<code>devmap-name</code>	Provides a meaningful device name to <code>udev</code> for device maps (devmaps).
<code>kpartx</code>	Maps linear devmaps to partitions on the multipathed device, which makes it possible to create multipath monitoring for partitions on the device.

For a list of files included in this package, see the [multipath-tools Package Description \(http://www.novell.com/products/linuxpackages/suselinux/multipath-tools.html\)](http://www.novell.com/products/linuxpackages/suselinux/multipath-tools.html).

Ensure that the `multipath-tools` package is installed by entering the following at a terminal console prompt:

```
rpm -q multipath-tools
```

5.2 Before You Begin

- ♦ [Section 5.2.1, “Preparing SAN Devices for Multipathing,” on page 44](#)
- ♦ [Section 5.2.2, “Partitioning Devices that Have Multiple Paths,” on page 45](#)
- ♦ [Section 5.2.3, “Configuring `mdadm.conf` and `lvm.conf` to Scan Devices by UUID,” on page 45](#)

5.2.1 Preparing SAN Devices for Multipathing

Before configuring multipath I/O for your SAN devices, prepare the SAN devices, as necessary, by doing the following:

- ♦ Configure and zone the SAN with the vendor’s tools.
- ♦ Configure permissions for host LUNs on the storage arrays with the vendor’s tools.
- ♦ Install the Linux HBA driver module. Upon module installation, the driver automatically scans the HBA to discover any SAN devices that have permissions for the host. It presents them to the host for further configuration.

NOTE: Ensure that the HBA driver you are using does not have native multipathing enabled.

See the vendor's specific instructions for more details.

- ♦ After the driver module is loaded, discover the device nodes assigned to specific array LUNs or partitions.

If the LUNs are not seen by the HBA driver, `lsscsi` can be used to check whether the SCSI devices are seen correctly by the operating system. When the LUNs are not seen by the HBA driver, check the zoning setup of the SAN. In particular, check whether LUN masking is active and whether the LUNs are correctly assigned to the server.

If the LUNs are seen by the HBA driver, but there are no corresponding block devices, additional kernel parameters are needed to change the SCSI device scanning behavior, such as to indicate that LUNs are not numbered consecutively. For information, see [Options for SCSI Device Scanning](http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=http--supportnovellcom-techcenter-sdb-en-2005-06-drahnscsiscanninghtml&sliceId=) (<http://www.novell.com/support/search.do?cmd=displayKC&docType=kc&externalId=http--supportnovellcom-techcenter-sdb-en-2005-06-drahnscsiscanninghtml&sliceId=>) in the Novell Technical Support Knowledgebase.

5.2.2 Partitioning Devices that Have Multiple Paths

Partitioning devices that have multiple paths is not recommended. However, if you want to partition the device, you should configure its partitions using `fdisk` or `YaST2` before configuring multipathing. This is necessary because partitioning a DM-MPIO device is not supported. Partitioning operations on `md#` devices fail if attempted.

If you configure partitions for a device, DM-MPIO automatically recognizes the partitions and indicates them by appending `p1-pn` to the device's UUID, such as

```
/dev/disk/by-id/3600601607cf30e00184589a37a31d911p1
```

To partition DM-MPIO devices, you must disable DM-MPIO, partition the normal device node (such as `/dev/sdc`), then reboot to allow DM-MPIO to see the new partitions.

5.2.3 Configuring `mdadm.conf` and `lvm.conf` to Scan Devices by UUID

Strange behavior occurs if you use the device node names (such as `/dev/sdc`) instead of the device's UUID. The UUID does not change by reboot or when a path is failed over. Multipathed devices show up automatically by their UUID in the `/dev/disk/by-id` directory.

The default settings in `mdadm.conf` and `lvm.conf` files do not work properly with multipathed devices. By default, both `md` and `LVM2` scan only the physical devices, and they ignore any symbolic links to devices and Device Mapper multipath I/O (DM-MPIO) devices. When managing DM-MPIO devices, you want the opposite behavior, so they ignore all physical devices, and scan only the devices listed in the `/dev/disk/by-id` directory.

To avoid problems, do the following:

- 1 In a terminal console, log in as the `root` user.
- 2 Open the `/etc/mdadm.conf` file in a text editor, then modify the `Devices` variable to scan for devices in the `/dev/disk/by-id` directory, as follows:

```
DEVICE /dev/disk/by-id/*
```

After you start multipath I/O services, the paths are listed under `/dev/disk/by-id` as these names are persistent. They are identical to their non-multipathed names.

5.3 Adding multipathd to the Boot Sequence

If you are using multipath IO services, add `multipathd` to the boot sequence, using one of the following methods:

- [Section 5.3.1, “YaST,” on page 46](#)
- [Section 5.3.2, “Command Line,” on page 46](#)

5.3.1 YaST

- 1 In YaST, click *System > System Services (Runlevel) > Simple Mode*.
- 2 Select *multipathd*, then click *Enable*.
- 3 Click *OK* to acknowledge the service startup message.
- 4 Click *Finish*, then click *Yes*.

The changes do not take effect until the server is restarted.

5.3.2 Command Line

- 1 Open a terminal console, then log in as the `root` user or equivalent.
- 2 At the terminal console prompt, enter

```
insserv boot.multipath multipathd
```

5.4 Starting Multipath I/O Services

To start multipath services and enable them to start at reboot:

- 1 Open a terminal console, then log in as the `root` user or equivalent.
- 2 At the terminal console prompt, enter

```
chkconfig multipathd on  
chkconfig boot.multipath on
```

If the `boot.multipath` service does not start automatically on system boot, do the following:

- 1 Open a terminal console, then log in as the `root` user or equivalent.
- 2 Enter

```
/etc/init.d/boot.multipath start  
/etc/init.d/multipathd start
```

5.5 Configuring Time-Out Settings for the HBA

When using multipath I/O, you want any host bus adapter (HBA) or cable failures to be reported faster than when single paths are used, so that multipathing can fail over to the healthy path. Configure time-out settings for your HBA to opt for failover sooner by modifying its options in the `/etc/modprobe.conf.local` file.

For example, for the QLogic 2xxx family of host bus adapters, the following setting are recommended:

```
options qla2xxx qlport_down_retry=1 ql2xretrycount=5
```

5.6 Configuring Multipath I/O for the Root Device

In the initial release, the root partition on multipath is supported only if the `/boot` partition is on a separate, non-multipathed partition. Otherwise, no bootloader is written.

NOTE: This issue is resolved in updates since the initial release.

To enable multipathing on the existing root device:

- 1 Install Linux with only a single path active, preferably one where the `by-id` symlinks are listed in the partitioner.
- 2 Mount the devices using the `/dev/disk/by-id` path used during the install.
- 3 After installation, add `dm-multipath` to `/etc/sysconfig/kernel:INITRD_MODULES`.
- 4 Re-run `/sbin/mkinitrd` to update `initrd` image.
- 5 Reboot the server.

To disable multipathing on the root device.

- 1 Add `multipath=off` to the kernel command line.

5.7 Scanning for New Devices without Rebooting

If your system has already been configured for multipathing and you later need to add more storage to the SAN, use the following procedure to scan the devices and make them available to multipathing without rebooting the system.

- 1 On the storage subsystem, use the vendor's tools to allocate the devices and update its access control settings to allow the Linux system access to the new storage. Refer to the vendor's documentation for details.
- 2 On the Linux system, use the HBA driver commands to scan the SAN to discover the new devices. The exact commands depend on the driver.

For example, for a QLogic 2300 HBA, the command is

```
echo scsi-qlascan >/proc/scsi/qla2xxx/<host number>
```

At this point, the newly added device is not known to the higher layers of the Linux kernel's SCSI subsystem and is not yet usable.

- 3 Scan all targets for a host to make its new device known to the middle layer of the Linux kernel's SCSI subsystem. At a terminal console prompt, enter

```
echo "- - -" >/sys/class/scsi_host/host<hostnumber>/scan
```
- 4 Run the Multipath tool to recognize the devices for DM-MPIO configuration. At a terminal console prompt, enter

```
multipath
```

You can now configure the devices for multipathing.

5.8 Configuring Multipathing for an Existing Software RAID

Ideally, you should configure multipathing for devices before you use them as components of a software RAID device. If you add multipathing after creating any software RAID devices, the Multipath service might be starting after the MD service on reboot, which makes multipathing appear not to be available for RAID. You can use the procedure in this section to get multipathing running for a previously existing software RAID.

For example, you might need to configure multipathing for devices in a software RAID under the following circumstances:

- ♦ If you create a new software RAID as part of the *Partitioning* settings during a new install or upgrade.
- ♦ If you did not configure the devices for multipathing before using them in the software RAID as a member device or spare.
- ♦ If you grow your system by adding new HBA adapters to the server or expanding the storage subsystem in your SAN.

NOTE: The following instructions assume the software RAID device is `/dev/md0`, which is its device name as recognized by the kernel. Make sure to modify the instructions for the device name of your software RAID.

- 1 Open a terminal console, then log in as the `root` user or equivalent.

Except where otherwise directed, use this console to enter the commands in the following steps.

- 2 If any software RAID devices are currently mounted or running, enter the following commands for each device to dismount the device and stop it.

```
umount /dev/md0
mdadm --misc --stop /dev/md0
```

- 3 Stop the `boot.md` service by entering

```
/etc/init.d/boot.md stop
```

- 4 Start the `boot.multipath` and `multipathd` services by entering the following commands:

```
/etc/init.d/boot.multipath start
/etc/init.s/multipathd start
```

- 5 After the multipathing services are started, verify that the software RAID's component devices are listed in the `/dev/disk/by-id` directory. Do one of the following:

- ♦ **Devices Are Listed:** The device names should now have symbolic links to their Device Mapper device names, such as `dev/dm-1`.
- ♦ **Devices Are Not Listed:** Force the `multipath` service to recognize them by flushing and rediscovering the devices.

To do this, enter the following commands:

```
multipath -F
multipath -v0
```


The devices should now be listed in `/dev/disk/by-id`, and have symbolic links to their Device Mapper device names. For example:

```
lrwxrwxrwx 1 root root 10 Jun 15 09:36 scsi-mpath1 -> ../../dm-1
```

- 6 Restart the `boot.md` service and the RAID device by entering

```
/etc/init.d/boot.md start
```

- 7 Check the status of the software RAID by entering

```
mdadm --detail /dev/md0
```

The RAID's component devices should match their Device Mapper device names that are listed as the symbolic links of devices in the `/dev/disk/by-id` directory.

- 8 Make a new `initrd` to ensure that the Device Mapper multipath services are loaded before the md RAID services on reboot. Enter

```
mkinitrd -f mpath
```

- 9 Reboot the server to apply these post-install configuration settings.

- 10 Verify that the software RAID array comes up properly on top of the multipathed devices by checking the RAID status. Enter

```
mdadm --detail /dev/md0
```

For example:

Number	Major	Minor	RaidDevice	State	
0	253	0	0	active sync	/dev/dm-0
1	253	1	1	active sync	/dev/dm-1
2	253	2	2	active sync	/dev/dm-2

5.9 Configuring User-Friendly Names in the `/etc/multipath.conf` File

The default name used in multipathing is the UUID of the logical unit as found in the `/dev/disk/by-id` directory. You can optionally override this behavior with user-friendly names instead. User-friendly names can be set via the `ALIAS` directive in the `multipath.conf` file.

IMPORTANT: We recommend that you do not use aliases for the root device as the ability to seamlessly switch off multipathing via the kernel command line is lost because the device names differ.

For an example of `multipath.conf` settings, see the `/usr/share/doc/packages/multipath-tools/multipath.conf.synthetic` file.

- 1 In a terminal console, log in as the `root` user.
- 2 Enter the following command (all on one line, of course):

```
cp /usr/share/doc/packages/multipath-tools/multipath.conf.synthetic /etc/multipath.conf
```
- 3 Open the `/etc/multipath.conf` file in a text editor.
- 4 Uncomment the `Defaults` directive and its ending bracket.
- 5 Uncomment the `user_friendly_names` option, then change its value from `No` to `Yes`.
- 6 Specify names for devices using the `ALIAS` directive.

- 7 Save your changes, then close the file.

5.10 Managing I/O in Error Situations

You might need to configure multipathing to queue I/O if all paths fail concurrently. In certain scenarios, where the driver, the HBA, or the fabric experiences spurious errors, it is advisable that DM MPIO be configured to queue all IO where those errors lead to a loss of all paths, and never propagate errors upwards. As this will lead to IO being queued forever, unless a path is reinstated, make sure that `multipathd` is running and works for your scenario. Otherwise, IO might be stalled forever on the affected MPIO device, until reboot or until you manually return to failover instead of queuing.

To test the scenario:

- 1 In a terminal console, log in as the `root` user.
- 2 Activate queuing instead of failover for the device I/O by entering (all on the same line):

```
dmsetup message 3600601607cf30e00184589a37a31d911 0  
queue_if_no_path
```

Replace the UUID (3600601607cf30e00184589a37a31d911) with the UUID for your device.
- 3 Return to failover for the device I/O by entering (all on the same line):

```
dmsetup message 3600601607cf30e00184589a37a31d911 0  
fail_if_no_path
```

Replace the UUID (3600601607cf30e00184589a37a31d911) with the UUID for your device.

This command immediately causes all queued I/O to fail.

To set up queuing I/O for scenarios where all paths fail:

- 1 In a terminal console, log in as the `root` user.
- 2 Open the `/etc/multipath.conf` file in a text editor.
- 3 Uncomment the defaults section and its ending bracket, then add the `default_features` setting, as follows:

```
defaults {  
default_features "1 queue_if_no_path"  
}
```
- 4 When you are ready to return over to failover for the device I/O, enter (all on the same line):

```
dmsetup message mapname 0 fail_if_no_path
```

Replace the *mapname* (such as 3600601607cf30e00184589a37a31d911) with the map name for the device.

This command immediately causes all queued I/O to fail and propagates the error to the calling application.

5.11 Resolving Stalled I/O

If all paths fail concurrently and I/O is queued and stalled, do the following:

- 1 Enter the following command at a terminal console prompt:

```
dmsetup message mapname 0 fail_if_no_path
```

Replace *mapname* with the correct map name, such as 3600601607cf30e00184589a37a31d911. This causes all queued I/O to fail and propagates the error to the calling application.

- 2 Reactivate queuing by entering the following command at a terminal console prompt:

```
dmsetup message mapname 0 queue_if_no_path
```

5.12 Additional Information

For more information about configuring and using multipath I/O on SUSE Linux Enterprise Server, see [How to Setup/Use Multipathing on SLES](http://support.novell.com/techcenter/sdb/en/2005/04/sles_multipathing.html) (http://support.novell.com/techcenter/sdb/en/2005/04/sles_multipathing.html) in the Novell Technical Support Knowledgebase.

5.13 What's Next

If you want to use software RAIDs, create and configure them before you create file systems on the devices. For information, see [Chapter 6, “Managing Software RAIDs with EVMS,”](#) on page 53.

Managing Software RAIDs with EVMS

6

This section discusses how to create and manage software RAIDs with the Enterprise Volume Management System (EVMS). EVMS supports only RAID 0, 1, 4, and 5 at this time. For RAID 6 and 10 solutions, see [Chapter 7, “Managing Software RAID 6 and 10 with mdadm,” on page 75](#).

- ♦ [Section 6.1, “Understanding Software RAID on Linux,” on page 53](#)
- ♦ [Section 6.2, “Creating and Configuring a Software RAID,” on page 59](#)
- ♦ [Section 6.3, “Expanding a RAID,” on page 63](#)
- ♦ [Section 6.4, “Adding or Removing a Spare Disk,” on page 64](#)
- ♦ [Section 6.5, “Managing Disk Failure and RAID Recovery,” on page 65](#)
- ♦ [Section 6.6, “Monitoring Status for a RAID,” on page 68](#)
- ♦ [Section 6.7, “Deleting a Software RAID and Its Data,” on page 73](#)

6.1 Understanding Software RAID on Linux

- ♦ [Section 6.1.1, “What Is a Software RAID?,” on page 53](#)
- ♦ [Section 6.1.2, “Overview of RAID Levels,” on page 54](#)
- ♦ [Section 6.1.3, “Comparison of RAID Performance,” on page 55](#)
- ♦ [Section 6.1.4, “Comparison of Disk Fault Tolerance,” on page 55](#)
- ♦ [Section 6.1.5, “Configuration Options for RAID,” on page 56](#)
- ♦ [Section 6.1.6, “Guidelines for Component Devices,” on page 56](#)
- ♦ [Section 6.1.7, “RAID 5 Algorithms for Distributing Stripes and Parity,” on page 57](#)
- ♦ [Section 6.1.8, “Multi-Disk Plug-In for EVMS,” on page 59](#)
- ♦ [Section 6.1.9, “Device Mapper Plug-In for EVMS,” on page 59](#)

6.1.1 What Is a Software RAID?

A RAID combines multiple devices into a multi-disk array to provide resiliency in the storage device and to improve storage capacity and I/O performance. If a disk fails, some RAID levels keep data available in a degraded mode until the failed disk can be replaced and its content reconstructed.

A software RAID provides the same high availability that you find in a hardware RAID. The key operational differences are described in the following table:

Table 6-1 *Comparison of Software RAID and Hardware RAID*

Feature	Linux Software RAID	Hardware RAID
RAID function	Multi-disk (<code>md</code>) driver or <code>mdadm</code>	RAID controller on the disk array

Feature	Linux Software RAID	Hardware RAID
RAID processing	In the host server's processor	RAID controller on the disk array
RAID levels	0, 1, 4, 5, and 10 plus the <code>mdadm</code> <code>raid10</code>	Varies by vendor
Component devices	Disks from same or different disk array	Same disk array

6.1.2 Overview of RAID Levels

The following table describes the advantages and disadvantages of the RAID levels supported by EVMS. The description assumes that the component devices reside on different disks and that each disk has its own dedicated I/O capability.

IMPORTANT: For information about creating complex or nested RAID devices with `mdadm`, see [Chapter 7, “Managing Software RAIDs 6 and 10 with `mdadm`,” on page 75](#).

Table 6-2 RAID Levels Supported by EVMS

RAID Level	Description	Performance and Fault Tolerance
0	Stripes data using a round-robin method to distribute data over the RAID's component devices.	Improves disk I/O performance for both reads and writes. Actual performance depends on the stripe size, the actual data, and the application. Does not provide disk fault tolerance and data redundancy. Any disk failure causes all data in the RAID to be lost.
1	Mirrors data by copying blocks of one disk to another and keeping them in continuous synchronization. If disks are different sizes, the smaller disk determines the size of the RAID.	Improves disk reads by making multiple copies of data available via different I/O paths. The write performance is about the same as for a single disk because a copy of the data must be written to each of the disks in the mirror. Provides 100% data redundancy. If one disk fails then the data remains available on its mirror, and processing continues.
4	Stripes data and distributes parity in a round-robin fashion across all disks. If disks are different sizes, the smaller disk determines the size of the RAID.	Improves disk I/O performance for reads and writes. Write performance is considerably less than for RAID 0, because parity must be calculated and written. Write performance is faster than RAID 4. Read performance is slower than for a RAID 1 array with the same number of component disks. Actual performance depends on the number of component disks, the stripe size, the actual data, and the application. Provides disk fault tolerance. If a disk fails, performance is degraded while the RAID uses the parity to reconstruct data for the replacement disk. Provides slightly less data redundancy than mirroring because it uses parity to reconstruct the data.

RAID Level	Description	Performance and Fault Tolerance
5	Stripes data and records parity to a dedicated disk. If disks are different sizes, the smaller disk determines the size of the RAID.	<p>Improves disk I/O performance for both reads and writes. Write performance is considerably slower than for RAID 0, because parity must be calculated and written. Write performance is slightly slower than RAID 5. Read performance is slower than for a RAID 1 array with the same number of component devices.</p> <p>This type of striping is seldom used because if the parity disk is lost, the parity data cannot be reconstructed. The parity disk can become a bottleneck for I/O.</p>

6.1.3 Comparison of RAID Performance

The following table compares the read and write performance for RAID devices.

Table 6-3 *Read and Write Performance for RAIDs*

Raid Level	Read Performance	Write Performance
0	Faster than for a single disk	Faster than for a single disk and other RAIDs.
1	Faster than for a single disk, increasing as more mirrors are added	Slower than for a single disk, declining as more mirrors are added.
4	Faster than for a single disk; comparable to a RAID 0	Faster than a single disk. Slower than a RAID 0 because of writes for parity.
5	Faster than for a single disk. Slower than a RAID 0 because one disk is used for parity	Faster than for a single disk. Slower than a RAID 0 because of writes for parity. Slower than a RAID 4 because of possible bottlenecks for writes of parity to the same disk.

6.1.4 Comparison of Disk Fault Tolerance

The following table compares the disk fault tolerance for RAID devices.

Table 6-4 *Fault Tolerance for RAIDs*

Raid Level	Number of Disk Failures Tolerated	Data Redundancy
0	None	No
1	Number of disks minus 1	100% redundancy for each mirror
4	1	Distributed parity to reconstruct data
5	1, but not the parity disk	Dedicated parity disk to reconstruct data

6.1.5 Configuration Options for RAIDs

Table 6-5 *Configuration Options*

Option	Description
Spare Disk	<p>For RAIDs 1, 4, and 5, you can optionally specify a device, segment, or region to use as the replacement for a failed disk (the member device, segment, or region). On failure, the spare disk automatically replaces the failed disk, then reconstructs the data.</p> <p>However, if the parity disk fails on a RAID 5, parity cannot be reconstructed.</p>
Chunk Size (KB)	<p>For RAIDs 0, 4, or 5, specify the stripe size in KB.</p> <p>Consider the intended use of the RAID, such as the file system block size, the applications used, and the actual data (file sizes and typical reads and writes). A typical write size for large files is 128 KB.</p> <p>Default: 32 KB</p> <p>Range: 4 KB to 4096 KB, in powers of 2.</p>
RAID Level	<p>If you selected <i>MD RAID 4/5 Region Manager</i>, specify <i>RAID 4</i> or <i>RAID 5</i> (default).</p>
RAID Algorithm	<p>For RAID 5, specify one of the following algorithms to use for striping and distributing parity on the disk.</p> <ul style="list-style-type: none">♦ Left Asymmetric♦ Left Symmetric (Default, fastest performance for large reads)♦ Right Asymmetric♦ Right Symmetric

6.1.6 Guidelines for Component Devices

For efficient use of space and performance, the disks you use to create the RAID should have the same storage capacity. Typically, if component devices are not of identical storage capacity, then each member of the RAID uses only an amount of space equal to the capacity of the smallest member disk.

Version 2.3 and later of `mdadm` supports component devices up to 4 TB in size each. Earlier versions support component devices up to 2 TB in size.

IMPORTANT: If you have a local disk, external disk arrays, or SAN devices that are larger than the supported device size, use a third-party disk partitioner to carve the devices into smaller logical devices.

You can combine up to 28 component devices to create the RAID array. The `md` RAID device you create can be up to the maximum device size supported by the file system you plan to use. For information about file system limits for SUSE[®] Linux Enterprise Server 10, see “Large File System Support” in the *SUSE Linux Enterprise Server 10 Installation and Administration Guide*. (<http://www.novell.com/documentation/sles10>).

In general, each storage object included in the RAID should be from a different physical disk to maximize I/O performance and to achieve disk fault tolerance where supported by the RAID level you use. In addition, they should be of the same type (disks, segments, or regions).

Using component devices of differing speeds might introduce a bottleneck during periods of demanding I/O. The best performance can be achieved by using the same brand and models of disks and controllers in your hardware solution. If they are different, you should try to match disks and controllers with similar technologies, performance, and capacity. Use a low number of drives on each controller to maximize throughput.

IMPORTANT: As with any hardware solution, using the same brand and model introduces the risk of concurrent failures over the life of the product, so plan maintenance accordingly.

The following table provides recommendations for the minimum and maximum number of storage objects to use when creating a software RAID:

Table 6-6 *Recommended Number of Storage Objects to Use in the Software RAID*

RAID Type	Minimum Number of Storage Objects	Recommended Maximum Number of Storage Objects
RAID 0 (striping)	2	8
RAID 1 (mirroring)	2	4
RAID 4 (striping with dedicated parity)	3	8
RAID 5 (striping with distributed parity)	3	8

Connection fault tolerance can be achieved by having multiple connection paths to each storage object in the RAID. For more information about configuring multipath I/O support before configuring a software RAID, see [Chapter 5, “Managing Multipath I/O for Devices,” on page 41](#).

6.1.7 RAID 5 Algorithms for Distributing Stripes and Parity

RAID 5 uses an algorithm to determine the layout of stripes and parity. The following table describes the algorithms.

Table 6-7 RAID 5 Algorithms

Algorithm	EVMS Type	Description
Left Asymmetric	1	<p>Stripes are written in a round-robin fashion from the first to last member segment. The parity's position in the striping sequence moves in a round-robin fashion from last to first. For example:</p> <pre> sda1 sdb1 sdc1 sde1 0 1 2 p 3 4 p 5 6 p 7 8 p 9 10 11 12 13 14 p </pre>
Left Symmetric	2	<p>This is the default setting and is considered the fastest method for large reads.</p> <p>Stripes wrap to follow the parity. The parity's position in the striping sequence moves in a round-robin fashion from last to first. For example:</p> <pre> sda1 sdb1 sdc1 sde1 0 1 2 p 4 5 p 3 8 p 6 7 p 9 10 11 12 13 14 p </pre>
Right Asymmetric	3	<p>Stripes are written in a round-robin fashion from the first to last member segment. The parity's position in the striping sequence moves in a round-robin fashion from first to last. For example:</p> <pre> sda1 sdb1 sdc1 sde1 p 0 1 2 3 p 4 5 6 7 p 8 9 10 11 p p 12 13 14 </pre>
Right Symmetric	4	<p>Stripes wrap to follow the parity. The parity's position in the striping sequence moves in a round-robin fashion from first to last. For example:</p> <pre> sda1 sdb1 sdc1 sde1 p 0 1 2 5 p 3 4 7 8 p 6 9 10 11 p p 12 13 14 </pre>

For information about the layout of stripes and parity with each of these algorithms, see [Linux RAID-5 Algorithms \(http://www.accs.com/p_and_p/RAID/LinuxRAID.html\)](http://www.accs.com/p_and_p/RAID/LinuxRAID.html).

6.1.8 Multi-Disk Plug-In for EVMS

The Multi-Disk (md) plug-in supports creating software RAIDs 0 (striping), 1 (mirror), 4 (striping with dedicated parity), and 5 (striping with distributed parity). The MD plug-in to EVMS allows you to manage all of these MD features as “regions” with the Regions Manager.

6.1.9 Device Mapper Plug-In for EVMS

The Device Mapper plug-in supports the following features in the EVMS MD Region Manager:

- ♦ **Multipath I/O:** Connection fault tolerance and load balancing for connections between the server and disks where multiple paths are available. If you plan to use multipathing, you should configure MPIO for the devices that you plan to use in the RAID before configuring the RAID itself. For information, see [Chapter 5, “Managing Multipath I/O for Devices,” on page 41](#).

IMPORTANT: The EVMS interface manages multipathing under the MD Region Manager, which originally supported the `md multipath` functions. It uses the legacy `md` terminology in the interface and in naming of device nodes, but implements the storage objects with Device Mapper.

- ♦ **Linear RAID:** A linear concatenation of discontinuous areas of free space from the same or multiple storage devices. Areas can be of different sizes.
- ♦ **Snapshots:** Snapshots of a file system at a particular point in time, even while the system is active, thereby allowing a consistent backup

The Device Mapper driver is not started by default in the rescue system.

- 1 Open a terminal console, then log in as the `root` user or equivalent.
- 2 Start the Device Mapper by entering the following at the terminal console prompt:

```
/etc/init.d/boot.device-mapper start
```

6.2 Creating and Configuring a Software RAID

- 1 Open a terminal console, then log in as the `root` user or equivalent.
- 2 Start the EVMS GUI by entering the following at the terminal console prompt:

```
evmsgui
```
- 3 If the disks have not been initialized, initialize them by adding the DOS Segment Manager now.

The following instructions assume you are initializing new disks. For information about initializing an existing disk or a disk moved from another system, see [Section 4.2, “Initializing Disks,” on page 34](#).

Repeat the following steps for each disk that you want to initialize:

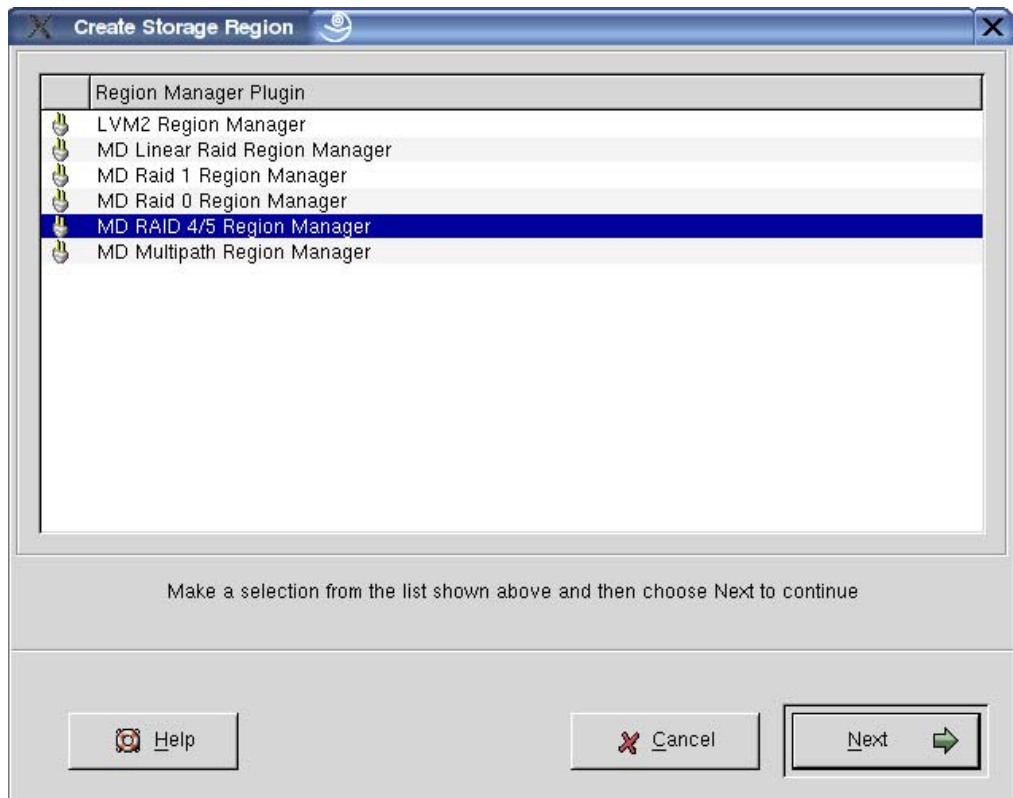
- 3a** Select *Actions > Add > Segment Manager to Storage Object*.
 - 3b** From the list, select the *DOS Segment Manager*, then click *Next*.
 - 3c** Select the device, then click *Add* to initialize it.
- 4 If segments have not been created on the disks, create a segment on each disk that you plan to use in the RAID.

For x86 platforms, this step is optional if you treat the entire disk as one segment.

For IA-64 platforms, this step is necessary to make the *RAID 4/5* option available in the Regions Manager.

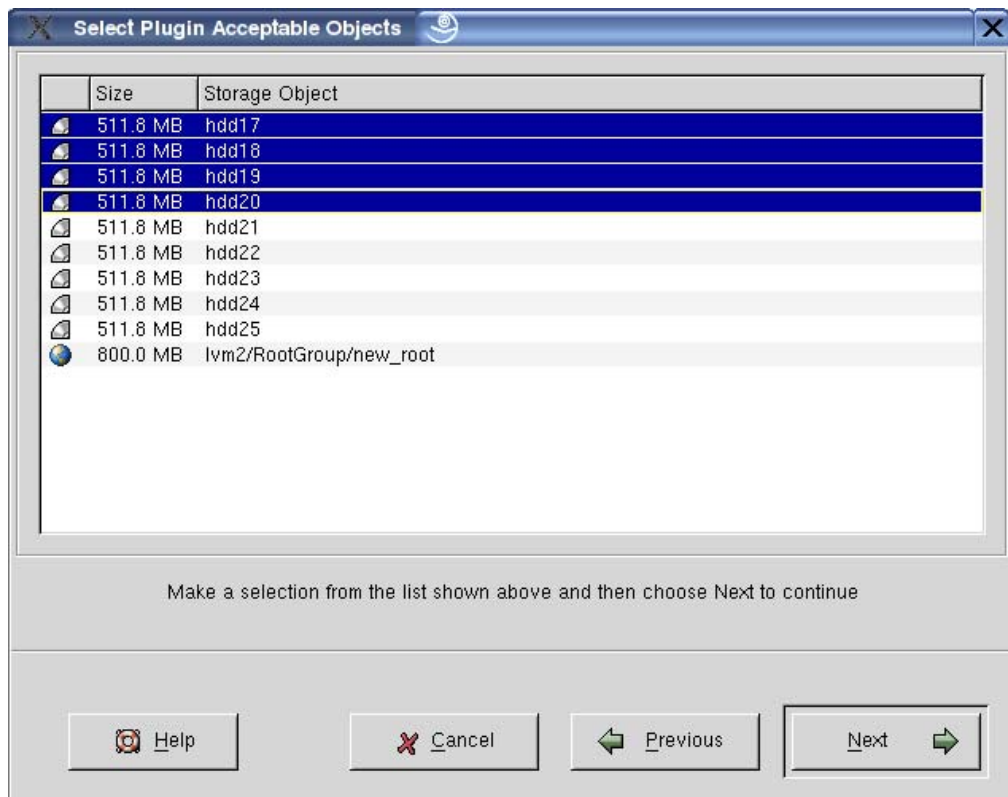
For information about creating segments, see [Section 4.4, “Creating Disk Segments \(or Partitions\),” on page 36](#).

- 4a** Select *Action > Create > Segment* to open the *DOS Segment Manager*.
 - 4b** Select the free space segment you want to use.
 - 4c** Specify the amount of space to use for the segment.
 - 4d** Specify the segment options, then click *Create*.
- 5** Create and configure a software RAID Device.
- 5a** Select *Action > Create > Region* to open the Create Storage Region dialog box.



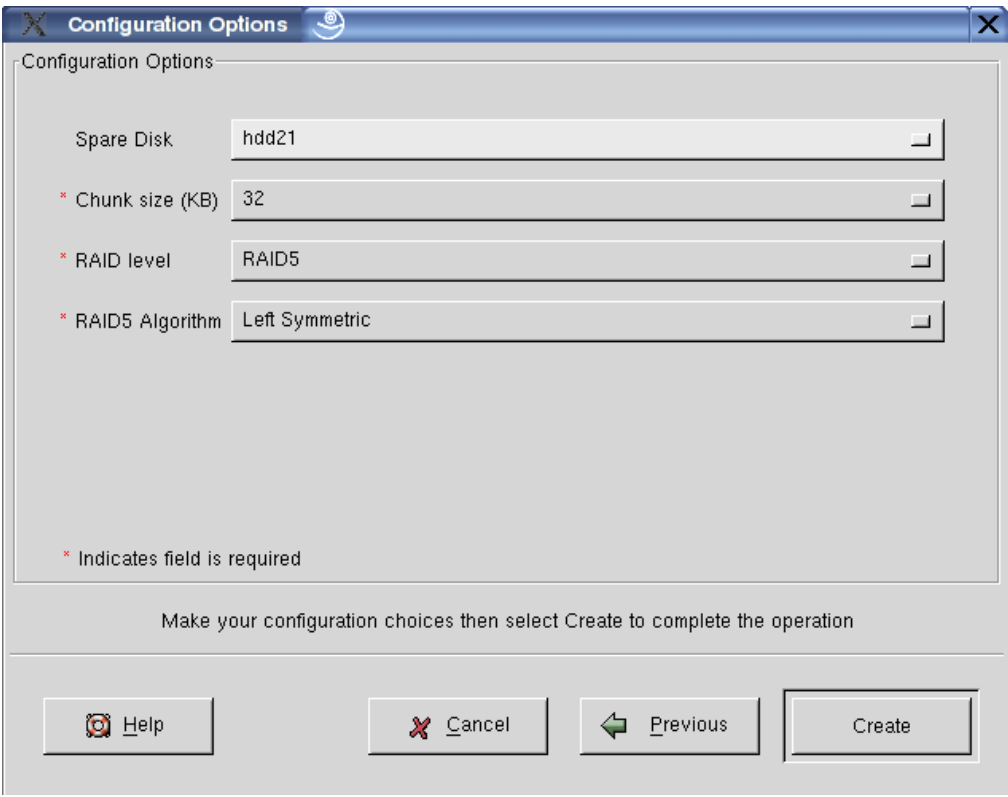
- 5b** Specify the type of software RAID you want to create by selecting one of the following Region Managers, then click *Next*.
 - ♦ *MD RAID 0 Region Manager*
 - ♦ *MD RAID 1 Region Manager*
 - ♦ *MD RAID 4/5 Region Manager*
- 5c** From the Storage Objects listed, select the ones to use for the RAID device.

IMPORTANT: The order of the objects in the RAID is implied by their order in the list.



- 5d** Specify values for *Configuration Options* by changing the following default settings as desired.
- For RAID 1, 4, or 5, optionally specify a device to use as the spare disk for the RAID. The default is none.
 - For RAID 0, 4, or 5, specify the chunk (stripe) size in KB. The default is 32 KB.
 - For RAID 4/5, specify *RAID 4* or *RAID 5* (default).
 - For RAID 5, specify the algorithm to use for striping and parity. The default is *Left Symmetric*.

For information about these settings, see “Configuration Options for RAID5” on page 56.



The screenshot shows a window titled "Configuration Options" with a close button (X) in the top right corner. The window contains several configuration fields:

- Spare Disk:** A text box containing "hdd21".
- * Chunk size (KB):** A text box containing "32".
- * RAID level:** A dropdown menu showing "RAID5".
- * RAID5 Algorithm:** A dropdown menu showing "Left Symmetric".

Below these fields is a note: "* Indicates field is required". At the bottom of the window, there is a message: "Make your configuration choices then select Create to complete the operation". Below this message are four buttons: "Help" (with a question mark icon), "Cancel" (with a red X icon), "Previous" (with a left arrow icon), and "Create".

- 5e** Click *Create* to create the RAID device under the `/dev/evms/md` directory.
The device is given a name such as `md0`, so its EVMS mount location is `/dev/evms/md/md0`.
- 6** Specify a human-readable label for the device.
 - 6a** Select *Action > Create > EVMS Volume or Compatible Volume*.
 - 6b** Select the device that you created in **Step 5**.
 - 6c** Specify a name for the device.
Use standard ASCII characters and naming conventions. Spaces are allowed.
 - 6d** Click *Done*.
- 7** Create a file system on the RAID device you created.
 - 7a** Select *Action > File System > Make* to view a list of file system modules.
 - 7b** Select the type of file system you want to create, such as the following:
 - ♦ *ReiserFS File System Module*
 - ♦ *Ext2/3FS File System Module*
 - 7c** Select the RAID device you created in **Step 5**, such as `/dev/evms/md/md0`.
 - 7d** Specify a name to use as the *Volume Label*, then click *Make*.
The name must not contain space or it will fail to mount later.
 - 7e** Click *Save* to create the file system.
- 8** Mount the RAID device.

- 8a** Select *Action > File System > Mount*.
- 8b** Select the RAID device you created in [Step 5](#), such as `/dev/evms/md/md0`.
- 8c** Specify the location where you want to mount the device, such as `/home`.
- 8d** Click *Mount*.
- 9** Enable `boot.evms` to activate EVMS automatically at reboot.
 - 9a** In YaST, select *System > System Services (Run Level)*.
 - 9b** Select *Expert Mode*.
 - 9c** Select *Boot.evms*.
 - 9d** Select *Set/Reset*.
 - 9e** Select *Enable the Service*.
- 10** Edit the `/etc/fstab` file to automount the RAID mount point created in [Step 8c](#), or you can mount the device manually from `evmsgui`.

6.3 Expanding a RAID

This section explains how to expand a RAID by adding segments to it.

IMPORTANT: Before you can expand the size of a RAID device, you must deactivate it.

- ♦ [Section 6.3.1, “Adding Mirrors to a RAID 1 Device,” on page 63](#)
- ♦ [Section 6.3.2, “Adding Segments to a RAID 4 or 5,” on page 64](#)

6.3.1 Adding Mirrors to a RAID 1 Device

In a RAID 1 device, each member segment contains its own copy of all of the data stored in the RAID. You can add a mirror to the RAID to increase redundancy. The segment must be at least the same size as the smallest member segment in the existing RAID 1 device. Any excess space in the segment is not used. Ideally, all member segments of a RAID 1 device are the same size.

Adding an Available Segment as the New Mirror

- 1** Deactivate the RAID 1 device.
- 2** Use the Add Active (`addactive` plug-in) function.
- 3** From the list of available segments, select one that is the same size or larger than the smallest existing member of the RAID device.
- 4** Reactivate the RAID device.

Activating A Spare Disk as the New Mirror

- 1** If you have not set up a spare disk, do it now.

For information, see [Section 6.4, “Adding or Removing a Spare Disk,” on page 64](#).
- 2** Use the Activate Spare (`activatespare` plug-in) function to add it to the RAID 1 device as a new mirror.

6.3.2 Adding Segments to a RAID 4 or 5

If the RAID region is clean and operating normally, the kernel driver adds the new object as a regular spare, and it acts as a hot standby for future failures. If the RAID region is currently degraded, the kernel driver immediately activates the new spare object and begin synchronizing the data and parity information.

6.4 Adding or Removing a Spare Disk

The MD driver allows you to optionally designate a spare disk (device, segment, or region) for RAID 1, 4, and 5 devices. You can assign a spare disk when you create the RAID or at any time thereafter. The RAID can be active and in use when you add or remove the spare. The spare is activated for the RAID only on disk failure.

- ♦ [Section 6.4.1, “Do I Need a Spare Disk?,” on page 64](#)
- ♦ [Section 6.4.2, “Adding a Spare Disk When You Create the RAID,” on page 65](#)
- ♦ [Section 6.4.3, “Adding a Spare Disk to an Existing RAID,” on page 65](#)
- ♦ [Section 6.4.4, “Removing a Spare Disk from a RAID,” on page 65](#)

6.4.1 Do I Need a Spare Disk?

The advantage of specifying a spare disk for a RAID is that the system monitors the failure and begins recovery without human interaction. The disadvantage is that the space on the spare disk is not available until it is activated by a failed RAID.

As noted in [“Overview of RAID Levels” on page 54](#), RAID 1, 4, and 5 can tolerate at least one disk failure. Any given RAID can have one spare disk designated for it, but the spare itself can serve as the designated spare for one RAID, for multiple RAID, or for all arrays. The spare disk is a hot standby until it is needed. It is not an active member of any RAID where it is assigned as the spare disk until it is activated for that purpose.

If a spare disk is defined for the RAID, the RAID automatically deactivates the failed disk and activates the spare disk on disk failure. The MD driver then begins synchronizing mirrored data for a RAID 1 or reconstructing the missing data and parity information for RAID 4 and 5. The I/O performance remains in a degraded state until the failed disk’s data is fully remirrored or reconstructed.

Creating a spare-group name allows a single hot spare to service multiple RAID arrays. The spare-group name can be any character string, but must be uniquely named for the server. For `mdadm` to move spares from one array to another, the different arrays must be labelled with the same spare-group name in the configuration file.

For example, when `mdadm` detects that an array is missing a component device, it first checks to see if the array has a spare device. If no spare is available, `mdadm` looks in the array’s assigned spare-group for another array that has a full complement of working drives and a spare. It attempts to remove the spare from the working array and add it to the degraded array. If the removal succeeds but the adding fails, then the spare is added back to its source array.

6.4.2 Adding a Spare Disk When You Create the RAID

When you create a RAID 1, 4, or 5 in EVMS, specify the *Spare Disk* in the *Configuration Options* dialog box. You can browse to select the available device, segment, or region that you want to make the RAID's spare disk. For information, see [Step 5d](#) in [Section 6.2, "Creating and Configuring a Software RAID,"](#) on page 59.

6.4.3 Adding a Spare Disk to an Existing RAID

The RAID 1, 4, or 5 device can be active and in use when you add a spare disk to it. If the RAID is operating normally, the specified disk is added as a spare and it acts as a hot standby for future failures. If the RAID is currently degraded because of a failed disk, the specified disk is added as a spare disk, then it is automatically activated as a replacement disk for the failed disk, and it begins synchronizing the data and parity information.

- 1 Prepare a disk, segment, or region to use as the replacement disk, just as you did for the component devices of the RAID device.
- 2 In EVMS, select the *Actions > Add > Spare Disk to a Region* (the `addspare` plug-in for the EVMS GUI).
- 3 Select the RAID device you want to manage from the list of Regions, then click *Next*.
- 4 Select the device to use as the spare disk.
- 5 Click *Add*.

6.4.4 Removing a Spare Disk from a RAID

The RAID 1, 4, or 5 device can be active and in use when you remove its spare disk.

- 1 In EVMS, select the *Actions > Remove > Spare Disk from a Region* (the `remspare` plug-in for the EVMS GUI).
- 2 Select the RAID device you want to manage from the list of Regions, then click *Next*.
- 3 Select the spare disk.
- 4 Click *Remove*.

6.5 Managing Disk Failure and RAID Recovery

- ♦ [Section 6.5.1, "Understanding the Disk Failure and RAID Recovery,"](#) on page 65
- ♦ [Section 6.5.2, "Identifying the Failed Drive,"](#) on page 66
- ♦ [Section 6.5.3, "Replacing a Failed Device with a Spare,"](#) on page 67
- ♦ [Section 6.5.4, "Removing the Failed Disk,"](#) on page 68

6.5.1 Understanding the Disk Failure and RAID Recovery

RAIDs 1, 4, and 5 can survive a disk failure. A RAID 1 device survives if all but one mirrored array fails. Its read performance is degraded without the multiple data sources available, but its write performance might actually improve when it does not write to the failed mirrors. During the synchronization of the replacement disk, write and read performance are both degraded. A RAID 5

can survive a single disk failure at a time. A RAID 4 can survive a single disk failure at a time if the disk is not the parity disk.

Disks can fail for many reasons such as the following:

- ♦ Disk crash
- ♦ Disk pulled from the system
- ♦ Drive cable removed or loose
- ♦ I/O errors

When a disk fails, the RAID removes the failed disk from membership in the RAID, and operates in a degraded mode until the failed disk is replaced by a spare. Degraded mode is resolved for a single disk failure in one of the following ways:

- ♦ **Spare Exists:** If the RAID has been assigned a spare disk, the MD driver automatically activates the spare disk as a member of the RAID, then the RAID begins synchronizing (RAID 1) or reconstructing (RAID 4 or 5) the missing data.
- ♦ **No Spare Exists:** If the RAID does not have a spare disk, the RAID operates in degraded mode until you configure and add a spare. When you add the spare, the MD driver detects the RAID's degraded mode, automatically activates the spare as a member of the RAID, then begins synchronizing (RAID 1) or reconstructing (RAID 4 or 5) the missing data.

6.5.2 Identifying the Failed Drive

On failure, md automatically removes the failed drive as a component device in the RAID array. To determine which device is a problem, use `mdadm` and look for the device that has been reported as "removed".

- 1 Enter the following at a terminal console prompt

```
mdadm -D /dev/md1
```

Replace `/dev/md1` with the actual path for your RAID.

For example, an `mdadm` report for a RAID 1 device consisting of `/dev/sda2` and `/dev/sdb2` might look like this:

```
blue6:~ # mdadm -D /dev/md1
/dev/md1:
    Version : 00.90.03
  Creation Time : Sun Jul  2 01:14:07 2006
    Raid Level : raid1
    Array Size : 180201024 (171.85 GiB 184.53 GB)
    Device Size : 180201024 (171.85 GiB 184.53 GB)
    Raid Devices : 2
    Total Devices : 1
Preferred Minor : 1
    Persistence : Superblock is persistent
    Update Time : Tue Aug 15 18:31:09 2006
        State : clean, degraded
    Active Devices : 1
    Working Devices : 1
```

```

Failed Devices : 0
Spare Devices : 0
    UUID : 8a9f3d46:3ec09d23:86e1ffbc:ee2d0dd8
    Events : 0.174164
    Number   Major   Minor   RaidDevice State
      0       0       0       0       removed
      1       8      18       1   active sync  /dev/sdb2

```

The “Total Devices : 1”, “Active Devices : 1”, and “Working Devices : 1” indicate that only one of the two devices is currently active. The RAID is operating in a “degraded” state.

The “Failed Devices : 0” might be confusing. This setting has a non-zero number only for that brief period where the md driver finds a problem on the drive and prepares to remove it from the RAID. Once the failed drive is removed, it reads “0” again.

In the devices list at the end of the report, the device with the “removed” state for Device 0 indicates that the device has been removed from the software RAID definition, not that the device has been physically removed from the system. It does not specifically identify the failed device. However, the working device (or devices) are listed. Hopefully, you have a record of which devices were members of the RAID. By the process of elimination, the failed device is /dev/sda2.

The “Spare Devices : 0” indicates that you do not have a spare assigned to the RAID. You must assign a spare device to the RAID so that it can be automatically added to the array and replace the failed device.

6.5.3 Replacing a Failed Device with a Spare

When a component device fails, the md driver replaces the failed device with a spare device assigned to the RAID. You can either keep a spare device assigned to the RAID as a hot standby to use as an automatic replacement, or assign a spare device to the RAID as needed.

IMPORTANT: Even if you correct the problem that caused the problem disk to fail, the RAID does not automatically accept it back into the array because it is a “faulty object” in the RAID and is no longer synchronized with the RAID.

If a spare is available, md automatically removes the failed disk, replaces it with the spare disk, then begins to synchronize the data (for RAID 1) or reconstruct the data from parity (for RAIDs 4 or 5).

If a spare is not available, the RAID operates in degraded mode until you assign spare device to the RAID.

To assign a spare device to the RAID:

- 1 Prepare the disk as needed to match the other members of the RAID.
- 2 In EVMS, select the *Actions > Add > Spare Disk to a Region* (the addspare plug-in for the EVMS GUI).
- 3 Select the RAID device you want to manage from the list of Regions, then click *Next*.
- 4 Select the device to use as the spare disk.
- 5 Click *Add*.

The md driver automatically begins the replacement and reconstruction or synchronization process.

- 6 Monitor the status of the RAID to verify the process has begun.
For information about how monitor RAID status, see [Section 6.6, “Monitoring Status for a RAID,” on page 68](#).
- 7 Continue with [Section 6.5.4, “Removing the Failed Disk,” on page 68](#).

6.5.4 Removing the Failed Disk

You can remove the failed disk at any time after it has been replaced with the spare disk. EVMS does not make the device available for other use until you remove it from the RAID. After you remove it, the disk appears in the *Available-Objects* list in the EVMS GUI, where it can be used for any purpose.

NOTE: If you pull a disk or if it is totally unusable, EVMS no longer recognizes the failed disk as part of the RAID.

The RAID device can be active and in use when you remove its faulty object.

- 1 In EVMS, select the *Actions > Remove > Faulty Object from a Region* (the `remfaulty` plug-in in the EVMS GUI).
- 2 Select the RAID device you want to manage from the list of Regions, then click *Next*.
- 3 Select the failed disk.
- 4 Click *Remove*.

6.6 Monitoring Status for a RAID

- ♦ [Section 6.6.1, “Monitoring Status with EVMSGUI,” on page 68](#)
- ♦ [Section 6.6.2, “Monitoring Status with /proc/mdstat,” on page 68](#)
- ♦ [Section 6.6.3, “Monitoring Status with mdadm,” on page 69](#)
- ♦ [Section 6.6.4, “Monitoring a Remirror or Reconstruction,” on page 71](#)
- ♦ [Section 6.6.5, “Configuring mdadm to Send an E-Mail Alert for RAID Events,” on page 71](#)

6.6.1 Monitoring Status with EVMSGUI

The *Regions* tab in EVMS GUI (`evmsgui`) reports any software RAID devices that are defined and whether they are currently active.

6.6.2 Monitoring Status with /proc/mdstat

A summary of RAID and status information (active/not active) is available in the `/proc/mdstat` file.

- 1 Open a terminal console, then log in as the `root` user or equivalent.
- 2 View the `/proc/mdstat` file by entering the following at the console prompt:

```
cat /proc/mdstat
```
- 3 Evaluate the information.

The following table shows an example output and how to interpret the information.

Status Information	Description	Interpretation
Personalities : [raid5] [raid4]	List of the RAIDs on the server by RAID label.	You have two RAIDs defined with labels of <code>raid5</code> and <code>raid4</code> .
md0 : active raid5 sdg1[0] sdk1[4] sdj1[3] sdi1[2]	<device> : <active not active> <RAID label you specified> < storage object> [RAID order]	The RAID is active and mounted at <code>/dev/evms/md/md0</code> . The RAID label is <code>raid5</code> . The active segments are <code>sdg1</code> , <code>sdi1</code> , <code>sdj1</code> , and <code>sdk1</code> , as ordered in the RAID. The RAID numbering of 0 to 4 indicates that the RAID has 5 segments, and the second segment [1] is missing from the list. Based on the segment names, the missing segment is <code>sdh1</code> .
35535360 blocks level 5, 128k chunk, algorithm 2 [5/4] [U_UUU]	<number of blocks> blocks level < 0 1 4 5 > <stripe size in KB> chunk algorithm <1 2 3 4 > [number of devices/number of working devices] [U-UUU]	If the block size on the server is 4 KB, the total size of the RAID (including parity) is 142 GB, with a data capacity of 113.7 GB. The stripe size is 128 KB. The RAID is using left symmetric. algorithm <1 2 3 4 > [number of devices/number of working devices] [U-UUU]
unused devices: <none>	All segments in the RAID are in use.	There are no spare devices available on the server.

6.6.3 Monitoring Status with mdadm

To view the RAID status with the `mdadm` command, enter the following at a terminal prompt:

```
mdadm -D /dev/mdx
```

Replace `mdx` with the RAID device number.

Example 1: A Disk Fails

In the following example, only 4 of the 5 devices in the RAID are active (`Raid Devices : 5`, `Total Devices : 4`). When it was created, the component devices in the device were numbered 0 to 5 and are ordered according to their alphabetic appearance in the list where they were chosen, such as `/dev/sdg1`, `/dev/sdh1`, `/dev/sdi1`, `/dev/sdj1`, and `/dev/sdk1`. From the pattern of filenames of the other devices, you determine that the device that was removed was named `/dev/sdh1`.

```
/dev/md0:
```

```

        Version : 00.90.03
    Creation Time : Sun Apr 16 11:37:05 2006
        Raid Level : raid5
        Array Size : 35535360 (33.89 GiB 36.39 GB)
        Device Size : 8883840 (8.47 GiB 9.10 GB)
        Raid Devices : 5
        Total Devices : 4
    Preferred Minor : 0
        Persistence : Superblock is persistent
        Update Time : Mon Apr 17 05:50:44 2006
            State : clean, degraded
        Active Devices : 4
    Working Devices : 4
    Failed Devices : 0
        Spare Devices : 0
            Layout : left-symmetric
        Chunk Size : 128K
            UUID : 2e686e87:1eb36d02:d3914df8:db197afe
            Events : 0.189
Number   Major   Minor   RaidDevice State
    0         8       97         0    active sync   /dev/sdg1
    1         0        0         1    removed
    2         8      129         2    active sync   /dev/sdi1
    3         8      145         3    active sync   /dev/sdj1
    4         8      161         4    active sync   /dev/sdk1

```

Example 2: Spare Disk Replaces the Failed Disk

In the following mdadm report, only 4 of the 5 disks are active and in good condition (Active Devices : 4, Working Devices : 5). The failed disk was automatically detected and removed from the RAID (Failed Devices: 0). The spare was activated as the replacement disk, and has assumed the diskname of the failed disk (/dev/sdh1). The faulty object (the failed disk that was removed from the RAID) is not identified in the report. The RAID is running in degraded mode (State : clean, degraded, recovering). The data is being rebuilt (spare rebuilding /dev/sdh1), and the process is 3% complete (Rebuild Status : 3% complete).

```

mdadm -D /dev/md0
    /dev/md0:
        Version : 00.90.03
    Creation Time : Sun Apr 16 11:37:05 2006
        Raid Level : raid5
        Array Size : 35535360 (33.89 GiB 36.39 GB)
        Device Size : 8883840 (8.47 GiB 9.10 GB)
        Raid Devices : 5
        Total Devices : 5
    Preferred Minor : 0

```

```

Persistence : Superblock is persistent
Update Time : Mon Apr 17 05:50:44 2006
State : clean, degraded, recovering
Active Devices : 4
Working Devices : 5
Failed Devices : 0
Spare Devices : 1
Layout : left-symmetric
Chunk Size : 128K
Rebuild Status : 3% complete
UUID : 2e686e87:1eb36d02:d3914df8:db197afe
Events : 0.189

```

Number	Major	Minor	RaidDevice	State	
0	8	97	0	active sync	/dev/sdg1
1	8	113	1	spare rebuilding	/dev/sdh1
2	8	129	2	active sync	/dev/sdi1
3	8	145	3	active sync	/dev/sdj1
4	8	161	4	active sync	/dev/sdk1

6.6.4 Monitoring a Remirror or Reconstruction

You can follow the progress of the synchronization or reconstruction process by examining the `/proc/mdstat` file.

You can control the speed of synchronization by setting parameters in the `/proc/sys/dev/raid/speed_limit_min` and `/proc/sys/dev/raid/speed_limit_max` files. To speed up the process, echo a larger number into the `speed_limit_min` file.

6.6.5 Configuring mdadm to Send an E-Mail Alert for RAID Events

You might want to configure the `mdadm` service to an e-mail alert for software RAID events. Monitoring is only meaningful for RAID 1, 4, 5, 6, 10 or multipath arrays as only these have missing, spare, or failed drives to monitor. RAID 0 and Linear RAID 0 do not provide fault tolerance so they have no interesting states to monitor.

The following table identifies RAID events and which events trigger e-mail alerts. All events cause the program to be run. The program is run with two or three arguments: the event name, the array device (such as `/dev/md1`), and possibly a second device. For Fail, Fail Spare, and Spare Active the second device is the relevant component device. For MoveSpare, the second device is the array that the spare was moved from.

Table 6-8 RAID Events in mdadm

RAID Event	Trigger E-Mail Alert	Description
Device Disappeared	No	An <code>md</code> array that was previously configured appears to no longer be configured. (syslog priority: Critical) If <code>mdadm</code> was told to monitor an array which is RAID0 or Linear, then it will report <code>DeviceDisappeared</code> with the extra information <code>Wrong-Level</code> . This is because RAID0 and Linear do not support the device-failed, hot-spare and resynchronize operations that are monitored.
Rebuild Started	No	An <code>md</code> array started reconstruction. (syslog priority: Warning)
Rebuild NN	No	Where NN is 20, 40, 60, or 80. This indicates the percent completed for the rebuild. (syslog priority: Warning)
Rebuild Finished	No	An <code>md</code> array that was rebuilding is no longer rebuilding, either because it finished normally or was aborted. (syslog priority: Warning)
Fail	Yes	An active component device of an array has been marked as faulty. (syslog priority: Critical)
Fail Spare	Yes	A spare component device that was being rebuilt to replace a faulty device has failed. (syslog priority: Critical)
Spare Active	No	A spare component device which was being rebuilt to replace a faulty device has been successfully rebuilt and has been made active. (syslog priority: Info)
New Array	No	A new <code>md</code> array has been detected in the <code>/proc/mdstat</code> file. (syslog priority: Info)
Degraded Array	Yes	A newly noticed array appears to be degraded. This message is not generated when <code>mdadm</code> notices a drive failure that causes degradation. It is generated only when <code>mdadm</code> notices that an array is degraded when it first sees the array. (syslog priority: Critical)
Move Spare	No	A spare drive has been moved from one array in a spare-group to another to allow a failed drive to be replaced. (syslog priority: Info)
Spares Missing	Yes	The <code>mdadm.conf</code> file indicates that an array should have a certain number of spare devices, but <code>mdadm</code> detects that the array has fewer than this number when it first sees the array. (syslog priority: Warning)
Test Message	Yes	An array was found at startup, and the <code>--test</code> flag was given. (syslog priority: Info)

- 1 At a terminal console, log in as the `root` user.
- 2 Edit the `/etc/mdadm/mdadm.conf` file to add your e-mail address for receiving alerts. For example, specify the `MAILADDR` value (using your own e-mail address, of course):

```

DEVICE partitions
ARRAY /dev/md0 level=raid1 num-devices=2
      UUID=1c661ae4:818165c3:3f7a4661:af475fda
      devices=/dev/sdb3,/dev/sdc3

```



```
MAILADDR yourname@example.com
```

The MAILADDR line gives an e-mail address that alerts should be sent to when mdadm is running in `--monitor` mode with the `--scan` option. There should be only one MAILADDR line in `mdadm.conf`, and it should have only one address.

- 3 Start mdadm monitoring by entering the following at the terminal console prompt:

```
mdadm --monitor --mail=yourname@example.com --delay=1800 /dev/md0
```

The `--monitor` option causes mdadm to periodically poll a number of md arrays and to report on any events noticed. mdadm never exits once it decides that there are arrays to be checked, so it should normally be run in the background.

In addition to reporting events in this mode, mdadm might move a spare drive from one array to another if they are in the same spare-group and if the destination array has a failed drive but no spares.

Listing the devices to monitor is optional. If any devices are listed on the command line, mdadm monitors only those devices. Otherwise, all arrays listed in the configuration file are monitored. Further, if `--scan` option is added in the command, then any other md devices that appear in `/proc/mdstat` are also monitored.

For more information about using mdadm, see the `mdadm(8)` and `mdadm.conf(5)` man pages.

- 4 To configure the `/etc/init.d/mdadm` service as a script:

```
suse:~ # egrep 'MAIL|RAIDDEVICE' /etc/sysconfig/mdadm
MDADM_MAIL="yourname@example.com"
MDADM_RAIDDEVICES="/dev/md0"
MDADM_SEND_MAIL_ON_START=no
suse:~ # chkconfig mdadm --list
mdadm      0:off  1:off  2:off  3:on   4:off  5:on  6:off
```

6.7 Deleting a Software RAID and Its Data

If you want to remove the prior multipath settings, deactivate the RAID, delete the data on the RAID, and release all resources used by the RAID, do the following:

- 1 If you want to keep the data stored on the software RAID device, make sure to back up the data to alternate media, using your normal backup procedures. Make sure the backup is good before proceeding.
- 2 Open a terminal console prompt as the `root` user or equivalent. Use this console to enter the commands described in the remaining steps.
- 3 Dismount the software RAID device by entering

```
umount <raid-device>
```
- 4 Stop the RAID device and its component devices by entering

```
mdadm --stop <raid-device>
mdadm --stop <member-devices>
```

For more information about using mdadm, please see the `mdadm(8)` man page.
- 5 Delete all data on the disk by literally overwriting the entire device with zeroes. Enter

```
mdadm --misc --zero-superblock <member-devices>
```
- 6 You must now reinitialize the disks for other uses, just as you would when adding a new disk to your system.

Managing Software RAID 6 and 10 with mdadm

7

This section discusses how to create software RAID 6 and 10 devices, using the Multiple Devices Administration (`mdadm(8)`) tool. You can also use `mdadm` to create RAID 0, 1, 4, and 5. The `mdadm` tool provides the functionality of legacy programs `mdtools` and `raidtools`.

- ♦ [Section 7.1, “Creating a RAID 6,” on page 75](#)
- ♦ [Section 7.2, “Creating Nested RAID 10 Devices with mdadm,” on page 76](#)
- ♦ [Section 7.3, “Creating a Complex RAID 10 with mdadm,” on page 79](#)
- ♦ [Section 7.4, “Creating a Degraded RAID Array,” on page 83](#)

7.1 Creating a RAID 6

- ♦ [Section 7.1.1, “Understanding RAID 6,” on page 75](#)
- ♦ [Section 7.1.2, “Creating a RAID 6,” on page 76](#)

7.1.1 Understanding RAID 6

RAID 6 is essentially an extension of RAID 5 that allows for additional fault tolerance by using a second independent distributed parity scheme (dual parity). Even if one of the hard disk drives fails during the data recovery process, the systems continues operational, with no data loss.

RAID6 provides for extremely high data fault tolerance by sustaining multiple simultaneous drive failures. It handles the loss of any two devices without data loss. Accordingly, it requires $N+2$ drives to store N drives worth of data. It requires a minimum of 4 devices.

The performance for RAID 6 is slightly lower but comparable to RAID 5 in normal mode and single disk failure mode. It is very slow in dual disk failure mode, however.

Table 7-1 *Comparison of RAID 5 and RAID 6*

Feature	RAID 5	RAID 6
Number of devices	$N+1$, minimum of 3	$N+2$, minimum of 4
Parity	Distributed, single	Distributed, dual
Performance	Medium impact on write and rebuild	More impact on sequential write than RAID 5
Fault-tolerance	Failure of one component device	Failure of two component devices

7.1.2 Creating a RAID 6

The following procedure creates a RAID 6 with four devices: `/dev/sda1`, `/dev/sdb1`, `/dev/sdc1`, and `/dev/sdd1`. Make sure to modify the procedure to use your actual device nodes.

- 1 Open a terminal console, then log in as the `root` user or equivalent.
- 2 Create 2 software RAID 0 devices, using two different devices for each RAID 0 device. At the command prompt, enter these two commands:

```
mdadm --create /dev/md0 --run --level=raid6 --chunk=128 --raid-devices=4 /dev/sdb1 /dev/sdc1 /dev/sdc1 /dev/sdd1
```

The default chunk size is 64 (KB).
- 3 Create a file system on the RAID 6 device `/dev/md0`, such as a Reiser file system (`reiserfs`). For example, at the command prompt, enter

```
mkfs.reiserfs /dev/md0
```

Modify the command if you want to use a different file system.
- 4 Edit the `/etc/mdadm.conf` file to add entries for the component devices and the RAID device `/dev/md0`.
- 5 Edit the `/etc/fstab` file to add an entry for the RAID 6 device `/dev/md0`.
- 6 Reboot the server.
The RAID 6 device is mounted to `/local`.
- 7 (Optional) Add a hot spare to service the RAID array. For example, at the command prompt enter:

```
mdadm /dev/md0 -a /dev/sde1
```

7.2 Creating Nested RAID 10 Devices with mdadm

- ♦ [Section 7.2.1, “Understanding Nested RAID Devices,” on page 76](#)
- ♦ [Section 7.2.2, “Creating Nested RAID 10 \(1+0\) with mdadm,” on page 77](#)
- ♦ [Section 7.2.3, “Creating Nested RAID 10 \(0+1\) with mdadm,” on page 78](#)

7.2.1 Understanding Nested RAID Devices

A nested RAID device consists of a RAID array that uses another RAID array as its basic element, instead of using physical disks. The goal of this configuration is to improve the performance and fault tolerance of the RAID.

Linux supports nesting of RAID 1 (mirroring) and RAID 0 (striping) arrays. Generally, this combination is referred to as RAID 10. To distinguish the order of the nesting, this document uses the following terminology:

- ♦ **RAID 1+0:** RAID 1 (mirror) arrays are built first, then combined to form a RAID 0 (stripe) array.
- ♦ **RAID 0+1:** RAID 0 (stripe) arrays are built first, then combined to form a RAID 1 (mirror) array.

The following table describes the advantages and disadvantages of RAID 10 nesting as 1+0 versus 0+1. It assumes that the storage objects you use reside on different disks, each with a dedicated I/O capability.

Table 7-2 RAID Levels Supported in EVMS

RAID Level	Description	Performance and Fault Tolerance
10 (1+0)	RAID 0 (stripe) built with RAID 1 (mirror) arrays	<p>RAID 1+0 provides high levels of I/O performance, data redundancy, and disk fault tolerance. Because each member device in the RAID 0 is mirrored individually, multiple disk failures can be tolerated and data remains available as long as the disks that fail are in different mirrors.</p> <p>You can optionally configure a spare for each underlying mirrored array, or configure a spare to serve a spare group that serves all mirrors.</p>
10 (0+1)	RAID 1 (mirror) built with RAID 0 (stripe) arrays	<p>RAID 0+1 provides high levels of I/O performance and data redundancy, but slightly less fault tolerance than a 1+0. If multiple disks fail on one side of the mirror, then the other mirror is available. However, if disks are lost concurrently on both sides of the mirror, all data is lost.</p> <p>Although this solution offers less disk fault tolerance than a 1+0 solution, If you need to perform maintenance or maintain the mirror on a different site, you can take an entire side of the mirror offline and still have a fully functional storage device. Also, if you lose the connection between the two sites, either site operates independently of the other. That is not true if you stripe the mirrored segments, because the mirrors are managed at a lower level.</p> <p>If a device fails, the mirror on that side fails because RAID 1 is not fault-tolerant. Create a new RAID 0 to replace the failed side, then resynchronize the mirrors.</p>

7.2.2 Creating Nested RAID 10 (1+0) with mdadm

A nested RAID 1+0 is built by creating two or more RAID 1 (mirror) devices, then using them as component devices in a RAID 0.

IMPORTANT: If you need to manage multiple connections to the devices, you must configure multipath I/O before configuring the RAID devices. For information, see [Chapter 5, “Managing Multipath I/O for Devices,” on page 41](#).

The following procedure uses the device names shown in the following table. Make sure to modify the device names with the names of your own devices.

Table 7-3 Scenario for Creating a RAID 10 (1+0) by Nesting

Devices	RAID 1 (mirror)	RAID 1+0 (striped mirrors)
/dev/sdb1	/dev/md0	/dev/md2
/dev/sdc1		
/dev/sdd1	/dev/md1	
/dev/sde1		

- 1 Open a terminal console, then log in as the root user or equivalent.
- 2 Create 2 software RAID 1 devices, using two different devices for each RAID 1 device. At the command prompt, enter these two commands:

```
mdadm --create /dev/md0 --run --level=1 --raid-devices=2 /dev/sdb1 /dev/sdc1
mdadm --create /dev/md1 --run --level=1 --raid-devices=2 /dev/sdd1 /dev/sde1
```
- 3 Create the nested RAID 1+0 device. At the command prompt, enter the following command using the software RAID 1 devices you created in [Step 2](#):

```
mdadm --create /dev/md2 --run --level=0 --chunk=64 --raid-devices=2 /dev/md0 /dev/md1
```

The default chunk size is 64 (KB).
- 4 Create a file system on the RAID 1+0 device /dev/md2, such as a Reiser file system (reiserfs). For example, at the command prompt, enter

```
mkfs.reiserfs /dev/md2
```

Modify the command if you want to use a different file system.
- 5 Edit the /etc/mdadm.conf file to add entries for the component devices and the RAID device /dev/md2.
- 6 Edit the /etc/fstab file to add an entry for the RAID 1+0 device /dev/md2.
- 7 Reboot the server.

The RAID 1+0 device is mounted to /local.
- 8 (Optional) Add hot spares to service the underlying RAID 1 mirrors.

For information, see [Section 6.4, “Adding or Removing a Spare Disk,”](#) on page 64.

7.2.3 Creating Nested RAID 10 (0+1) with mdadm

A nested RAID 0+1 is built by creating two to four RAID 0 (striping) devices, then mirroring them as component devices in a RAID 1.

IMPORTANT: If you need to manage multiple connections to the devices, you must configure multipath I/O before configuring the RAID devices. For information, see [Chapter 5, “Managing Multipath I/O for Devices,”](#) on page 41.

In this configuration, spare devices cannot be specified for the underlying RAID 0 devices because RAID 0 cannot tolerate a device loss. If a device fails on one side of the mirror, you must create a replacement RAID 0 device, then add it into the mirror.

The following procedure uses the device names shown in the following table. Make sure to modify the device names with the names of your own devices.

Table 7-4 Scenario for Creating a RAID 10 (0+1) by Nesting

Devices	RAID 0 (stripe)	RAID 0+1 (mirrored stripes)
/dev/sdb1	/dev/md0	/dev/md2
/dev/sdc1		
/dev/sdd1	/dev/md1	
/dev/sde1		

- 1 Open a terminal console, then log in as the root user or equivalent.
 - 2 Create 2 software RAID 0 devices, using two different devices for each RAID 0 device. At the command prompt, enter these two commands:

```
mdadm --create /dev/md0 --run --level=0 --chunk=64 --raid-devices=2 /dev/sdb1 /dev/sdc1
```

```
mdadm --create /dev/md1 --run --level=0 --chunk=64 --raid-devices=2 /dev/sdd1 /dev/sde1
```

The default chunk size is 64 (KB).
 - 3 Create the nested RAID 0+1 device. At the command prompt, enter the following command using the software RAID 0 devices you created in [Step 2](#):

```
mdadm --create /dev/md2 --run --level=1 --raid-devices=2 /dev/md0 /dev/md1
```
 - 4 Create a file system on the RAID 0+1 device /dev/md2, such as a Reiser file system (reiserfs). For example, at the command prompt, enter

```
mkfs.reiserfs /dev/md2
```

Modify the command if you want to use a different file system.
 - 5 Edit the /etc/mdadm.conf file to add entries for the component devices and the RAID device /dev/md2.
 - 6 Edit the /etc/fstab file to add an entry for the RAID 0+1 device /dev/md2.
 - 7 Reboot the server.
- The RAID 0+1 device is mounted to /local.

7.3 Creating a Complex RAID 10 with mdadm

- ♦ [Section 7.3.1, “Understanding the mdadm RAID10,” on page 80](#)
- ♦ [Section 7.3.2, “Creating a RAID10 with mdadm,” on page 82](#)

7.3.1 Understanding the mdadm RAID10

In mdadm, the RAID10 level creates a single complex software RAID that combines features of both RAID 0 (striping) and RAID 1 (mirroring). Multiple copies of all data blocks are arranged on multiple drives following a striping discipline. Component devices should be the same size.

- ♦ “Comparison of RAID10 Option and Nested RAID 10 (1+0)” on page 80
- ♦ “Number of Replicas in the mdadm RAID10” on page 80
- ♦ “Number of Devices in the mdadm RAID10” on page 80
- ♦ “Near Layout” on page 81
- ♦ “Far Layout” on page 81

Comparison of RAID10 Option and Nested RAID 10 (1+0)

The complex RAID 10 is similar in purpose to a nested RAID 10 (1+0), but differs in the following ways:

Table 7-5 *Complex vs. Nested RAID 10*

Feature	mdadm RAID10 Option	Nested RAID 10 (1+0)
Number of devices	Allows an even or odd number of component devices	Requires an even number of component devices
Component devices	Managed as a single RAID device	Manage as a nested RAID device
Striping	Striping occurs in the near or far layout on component devices. The far layout provides sequential read throughput that scales by number of drives, rather than number of RAID 1 pairs.	Striping occurs consecutively across component devices
Multiple copies of data	Two or more copies, up to the number of devices in the array	Copies on each mirrored segment
Hot spare devices	A single spare can service all component devices	Configure a spare for each underlying mirrored array, or configure a spare to serve a spare group that serves all mirrors.

Number of Replicas in the mdadm RAID10

When configuring a RAID10-level array, you must specify the number of replicas of each data block that are required. The default number of replicas is 2, but the value can be 2 to the number of devices in the array.

Number of Devices in the mdadm RAID10

You must use at least as many component devices as the number of replicas you specify. However, number of component devices in a RAID10-level array does not need to be a multiple of the number

of replicas of each data block. The effective storage size is the number of devices divided by the number of replicas.

For example, if you specify 2 replicas for an array created with 5 component devices, a copy of each block is stored on two different devices. The effective storage size for one copy of all data is 5/2 or 2.5 times the size of a component device.

Near Layout

With the near layout, copies of a block of data are striped near each other on different component devices. That is, multiple copies of one data block are at similar offsets in different devices. Near is the default layout for RAID10. For example, if you use an odd number of component devices and two copies of data, some copies are perhaps one chunk further into the device.

The near layout for the `mdadm` RAID10 yields read and write performance similar to RAID 0 over half the number of drives.

Near layout with an even number of disks and two replicas:

sda1	sdb1	sdc1	sde1
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7
8	8	9	9

Near layout with an odd number of disks and two replicas:

sda1	sdb1	sdc1	sde1	sdf1
0	0	1	1	2
2	3	3	4	4
5	5	6	6	7
7	8	8	9	9
10	10	11	11	12

Far Layout

The far layout stripes data over the early part of all drives, then stripes a second copy of the data over the later part of all drives, making sure that all copies of a block are on different drives. The second set of values start half-way through the component drives.

With a far layout, the read performance of the `mdadm` RAID10 is similar to a RAID 0 over the full number of drives, but write performance is substantially slower than a RAID 0 because there is more seeking of the drive heads. It is best used for read-intensive operations such as for read-only file servers.

Far layout with an even number of disks and two replicas:

sda1	sdb1	sdc1	sde1
0	1	2	3
3	5	6	7
.	.	.	.

3	1	2	3
7	4	5	6

Far layout with an odd number of disks and two replicas:

sda1	sdb1	sdc1	sde1	sdf1
0	1	2	3	4
5	6	7	8	9
.	.	.		
4	0	1	2	3
9	5	6	7	8

7.3.2 Creating a RAID10 with mdadm

The RAID10-level option for mdadm creates a RAID 10 device without nesting. For information about the RAID10-level, see [Section 7.3, “Creating a Complex RAID 10 with mdadm,” on page 79](#).

The following procedure uses the device names shown in the following table. Make sure to modify the device names with the names of your own devices.

Table 7-6 Scenario for Creating a RAID 10 Using the mdadm RAID10 Option

Devices	RAID10 (near or far striping scheme)
/dev/sdf1	/dev/md3
/dev/sdg1	
/dev/sdh1	
/dev/sdi1	

- 1 In YaST, create a 0xFD Linux RAID partition on the devices you want to use in the RAID, such as /dev/sdf1, /dev/sdg1, /dev/sdh1, and /dev/sdi1.
- 2 Open a terminal console, then log in as the root user or equivalent.
- 3 Create a RAID 10 command. At the command prompt, enter (all on the same line):
mdadm --create /dev/md3 --run --level=10 --chunk=4 --raid-devices=4
/dev/sdf1 /dev/sdg1 /dev/sdh1 /dev/sdi1
- 4 Create a Reiser file system on the RAID 10 device /dev/md3. At the command prompt, enter
mkfs.reiserfs /dev/md3
- 5 Edit the /etc/mdadm.conf file to add entries for the component devices and the RAID device /dev/md3. For example:
DEVICE /dev/md3
- 6 Edit the /etc/fstab file to add an entry for the RAID 10 device /dev/md3.
- 7 Reboot the server.
The RAID10 device is mounted to /raid10.

7.4 Creating a Degraded RAID Array

A degraded array is one in which some devices are missing. Degraded arrays are supported only for RAID 1, RAID 4, RAID 5, and RAID 6. These RAID types are designed to withstand some missing devices as part of their fault-tolerance features. Typically, degraded arrays occur when a device fails. It is possible to create a degraded array on purpose.

RAID Type	Allowable Number of Slots Missing
RAID 1	All but one device
RAID 4	One slot
RAID 5	One slot
RAID 6	One or two slots

To create a degraded array in which some devices are missing, simply give the word `missing` in place of a device name. This causes `mdadm` to leave the corresponding slot in the array empty.

When creating a RAID 5 array, `mdadm` automatically creates a degraded array with an extra spare drive. This is because building the spare into a degraded array is generally faster than resynchronizing the parity on a non-degraded, but not clean, array. This feature can be over-ridden with the `--force` option.

Creating a degraded array might be useful if you want create a RAID, but one of the devices you want to use already has data on it. In that case, you create a degraded array with other devices, copy data from the in-use device to the RAID that is running in degraded mode, add the device into the RAID, then wait while the RAID is rebuilt so that the data is now across all devices. An example of this process is demonstrated in the following procedure:

- 1 Create a degraded RAID 1 device `/dev/md0`, using one single drive `/dev/sd1`, enter the following at the command prompt:

```
mdadm --create /dev/md0 -l 1 -n 2 /dev/sda1 missing
```

The device should be the same size or larger than the device you plan to add to it.

- 2 If the device you want to add to the mirror contains data that you want to move to the RAID array, copy it now to the RAID array while it is running in degraded mode.
- 3 Add a device to the mirror. For example, to add `/dev/sdb1` to the RAID, enter the following at the command prompt:

```
mdadm /dev/md0 -a /dev/sdb1
```

You can add only one device at a time. You must wait for the kernel to build the mirror and bring it fully online before you add another mirror.

- 4 Monitor the build progress, by entering the following at the command prompt:

```
cat /proc/mdstat
```


Installing and Managing DRBD Services

8

This section describes how to install, configure, and manage a device-level software RAID 1 across a network using DRBD* (Distributed Replicated Block Device) for Linux.

- ♦ [Section 8.1, “Understanding DRBD,” on page 85](#)
- ♦ [Section 8.2, “Installing DRBD Services,” on page 85](#)
- ♦ [Section 8.3, “Configuring the DRBD Service,” on page 86](#)

8.1 Understanding DRBD

DRBD allows you to create a mirror of devices that are located at two different sites across a network. When used with HeartBeat 2 (HB2), DRBD supports distributed high-availability Linux clusters.

Data on the primary device is replicated to the secondary device in a way that ensures that both copies of the data are always identical.

The data traffic between mirrors is not encrypted. For secure data exchange, you should deploy a virtual private network (VPN) solution for the connection.

8.1.1 Additional Information

The following open-source resources are available for DRBD:

- ♦ [DRBD.org \(http://www.drbd.org\)](http://www.drbd.org)
- ♦ [DRBD references at the Linux High-Availability Project \(http://linux-ha.org/DRBD\)](http://linux-ha.org/DRBD) by the Linux High-Availability Project

For information about installing and configuring HeartBeat 2 for SUSE® Linux Enterprise Server 10, see the *HeartBeat 2 Installation and Setup Guide* (http://www.novell.com/documentation/sles10/hb2/data/hb2_config.html) on the [Novell Documentation Web site for SUSE Linux Enterprise Server 10 \(http://www.novell.com/documentation/sles10\)](http://www.novell.com/documentation/sles10).

8.2 Installing DRBD Services

- 1 Install the High Availability (HA) pattern on both SUSE Linux Enterprise Servers in your networked cluster. Installing HA also installs the `drbd` program files.
 - 1a Log in as the `root` user or equivalent, then open YaST.
 - 1b Choose *Software > Software Management*.
 - 1c Change the filter to *Patterns*.
 - 1d Under *Base Technologies*, select *High Availability*.
 - 1e Click *Accept*.
- 2 Install the `drbd` kernel modules on both servers.

- 2a** Log in as the `root` user or equivalent, then open YaST.
- 2b** Choose *Software > Software Management*.
- 2c** Change the filter to *Search*.
- 2d** Type `drbd`, then click *Search*.
- 2e** Select all of the `drbd-kmp-*` packages.
- 2f** Click *Accept*.

8.3 Configuring the DRBD Service

NOTE: The following procedure uses the server names node 1 and node 2, and the cluster resource name `r0`. It sets up node 1 as the primary node. Make sure to modify the instructions to use your own node and file names.

- 1** Open the `/etc/drbd.conf` file on the primary node (node 1) in a text editor, modify the following parameters in the `on hostname { }` sections, then save the file.

- ♦ `hostname`
- ♦ `device`
- ♦ `disk`
- ♦ `meta-disk`
- ♦ `address`
- ♦ `port`

All of these options are explained in the examples in the `/usr/share/doc/packages/drbd/drbd.conf` file.

- 2** Copy the `/etc/drbd.conf` file to the `/etc/drbd.conf` location on the secondary server (node 2).
- 3** Configure the DRBD service for node 1.
 - 3a** Open a terminal console for node 1, then log in as the `root` user or equivalent.
 - 3b** Initialize DRBD on node 1 by entering

```
modprobe drbd
```
 - 3c** Test the configuration file by running `drbdadm` with the `-d` option. Enter

```
drbdadm -d adjust r0
```
 - 3d** If the partitions and settings are correct, run `drbdadm` again without the `-d` option. Enter

```
drbdadm adjust r0
```

Make sure there are no errors before continuing.
- 4** Configure the DRBD service for node 2.
 - 4a** Open a terminal console for node 2, then log in as the `root` user or equivalent.
 - 4b** Initialize the DRBD service on node 2 by entering

```
modprobe drbd
```
 - 4c** Test the configuration file by running `drbdadm` with the `-d` option. Enter

```
drbdadm -d adjust r0
```
 - 4d** If the partitions and settings are correct, run `drbdadm` again without the `-d` option. Enter

```
drbdadm adjust r0
```

Make sure there are no errors before continuing.

- 5** Configure node1 as the primary node by entering

```
drbdsetup /dev/drbd0 primary -do-what-I-say
```

- 6** Start the DRBD service on both systems by entering the following on each node:

```
service drbd start
```

- 7** Check the DRBD service status by entering the following on each node:

```
service drbd status
```

- 8** Format the DRBD device on the primary with a file system such as reiserfs. Any Linux file system can be used. Enter the following:

```
mkfs.reiserfs -f /dev/drbd0
```

Always use the /dev/drbd# name in the command, not the actual /dev/disk device name.

- 9** Test the DRBD service on node 1.

- 9a** Create a mount point on node 1, such as /r0mount, by entering

```
mkdir /r0mount
```

- 9b** Mount the drbd device by entering

```
mount -o rw /dev/drbd0 /r0mount
```

- 9c** Create a file from the primary node by entering

```
touch /r0mount/from_node1
```

- 10** Test the DRBD service on node 2.

- 10a** Dismount the disk on node 1 by typing the following command on node 1:

```
umount /r0mount
```

- 10b** Downgrade the DRBD service on node 1 by typing the following command on node 1:

```
drbdadm secondary r0
```

- 10c** On node 2, promote the DRBD service to primary by entering

```
drbdadm primary r0
```

- 10d** On node 2, check to see if node 2 is primary by entering

```
service drbd status
```

- 10e** On node 2, create a mount point such as /r0mount, by entering

```
mkdir /r0mount
```

- 10f** On node 2, mount the DRBD device by entering

```
mount -o rw /dev/drbd0 /r0mount
```

- 10g** Verify that the file you created on node 1 in [Step 9c](#) is viewable by entering

```
ls /r0mount
```

The /r0mount/from_node1 file should be listed.

- 11** If the service is working on both nodes, the DRBD setup is complete.

- 12** Set up node 1 as the primary again.

- 12a** Dismount the disk on node 2 by typing the following command on node 2:

```
umount /r0mount
```

- 12b** Downgrade the DRBD service on node 2 by typing the following command on node 2:

```
drbdadm secondary r0
```

12c On node 1, promote the DRBD service to primary by entering

```
drbdadm primary r0
```

12d On node 1, check to see if node 1 is primary by entering

```
service drbd status
```

13 To get the service to automatically start and fail over if the server has a problem, you can set up DRBD as a high availability service with HeartBeat 2.

Troubleshooting EVMS Devices, RAIDs, and Volumes

9

This section describes how to work around known issues for EVMS devices, software RAIDs, multipath I/O, and volumes.

- ♦ [Section 9.1, “EVMS Volumes Might Not Appear When Using iSCSI,” on page 89](#)
- ♦ [Section 9.2, “Device Nodes Are Not Automatically Re-Created on Restart,” on page 89](#)

9.1 EVMS Volumes Might Not Appear When Using iSCSI

If you have installed and configured an iSCSI SAN, and have created and configured EVMS Disks or Volumes on that iSCSI SAN, your EVMS volumes might not be visible or accessible. This problem is caused by EVMS starting before the iSCSI service. iSCSI must be started and running before any disks/volumes on the iSCSI SAN can be accessed.

To resolve this problem, use the `chkconfig` command at the Linux server console of every server that is part of your iSCSI SAN to correct the order that iSCSI and EVMS are started.

- 1 At a terminal console prompt, enter
`chkconfig boot.evms on`

This ensures that EVMS and iSCSI start in the proper order each time your servers reboot.

9.2 Device Nodes Are Not Automatically Re-Created on Restart

Effective in SUSE Linux Enterprise 10, the `/dev` directory is on `tmpfs` and the device nodes are automatically re-created on boot. It is no longer necessary to modify the `/etc/init.d/boot.evms` script to delete the device nodes on system reboot as was required for previous versions of SUSE Linux.

The following procedure is provided for users who might encounter this issue for SUSE Linux Enterprise Server 9 and earlier:

- 1 Open the `/etc/init.d/boot.evms` script in a text editor.
- 2 Add the following lines to the Stop section:

```
mount -n -o remount,rw /
echo -en "\nDeleting devices nodes"
rm -rf /dev/evms
mount -n -o remount,ro /
```

For example, the Stop section looks like this after the edit:

```
stop)
    echo -n "Stopping EVMS"
    mount -n -o remount,rw /
```

```
echo -en "\nDeleting devices nodes"
rm -rf /dev/evms
mount -n -o remount,ro /
rc_status -v
;;
```

- 3** Save the file.
- 4** Continue with **“Reboot the Server”** on page 21.

Documentation Updates

A

This section contains information about documentation content changes made to the *SUSE Linux Enterprise Server Storage Administration Guide for EVMS* since the initial release of SUSE® Linux Enterprise Server 10. If you are an existing user, review the change entries to readily identify modified content. If you are a new user, simply read the guide in its current state.

Refer to the publication date, which appears on the title page, to determine the release date of this guide. For the most recent version of the *SUSE Linux Enterprise Server Storage Administration Guide for EVMS*, see the [Novell documentation Web site for SUSE Linux Enterprise Server 10](http://www.novell.com/documentation/sles10) (<http://www.novell.com/documentation/sles10>).

In this section, content changes appear in reverse chronological order, according to the publication date. Within a dated entry, changes are grouped and sequenced, according to where they appear in the document itself. Each change entry provides a link to the related topic and a brief description of the change.

This document was updated on the following dates:

- ♦ [Section A.1, “February 1, 2007 \(Updates\),” on page 91](#)
- ♦ [Section A.2, “December 1, 2006 \(Updates\),” on page 92](#)

A.1 February 1, 2007 (Updates)

Updates were made to the following sections. The changes are explained below.

- ♦ [Section A.1.1, “Managing Software RAIDs with mdadm,” on page 91](#)

A.1.1 Managing Software RAIDs with mdadm

The following change was made to this section:

Location	Change
Section 7.1, “Creating a RAID 6,” on page 75	This section is new.
Section 7.2.1, “Understanding Nested RAID Devices,” on page 76	Changes were made for clarity.
Section 7.3.1, “Understanding the mdadm RAID10,” on page 80	Changes were made for clarity.
Section 7.4, “Creating a Degraded RAID Array,” on page 83	This section is new.

A.2 December 1, 2006 (Updates)

Updates were made to the following sections. The changes are explained below.

- ♦ Section A.2.1, “Overview of EVMS,” on page 92
- ♦ Section A.2.2, “Using EVMS to Manage Devices,” on page 92
- ♦ Section A.2.3, “Managing Multipathing for Devices and Software RAIDs,” on page 92
- ♦ Section A.2.4, “Managing Software RAIDs,” on page 93
- ♦ Section A.2.5, “Managing Software RAIDs with mdadm,” on page 93

A.2.1 Overview of EVMS

The following changes were made to this section:

Location	Change
Section 1.4, “Terminology,” on page 12	Changes were made for clarification.
Section 1.5, “Location of Device Nodes for EVMS Storage Objects,” on page 13	Changes were made for clarification.

A.2.2 Using EVMS to Manage Devices

The following changes were made to this section:

Location	Change
Section 2.2.3, “Edit the /etc/init.d/boot.evms Script,” on page 23	Effective in SUSE Linux Enterprise Server 10, this procedure is no longer necessary.
Section 2.2.5, “Force the RAM Disk to Recognize the Root Partition,” on page 25	Recent patches to <code>mkinitrd</code> might resolve the need to do this task. For the latest version of <code>mkinitrd</code> , see Recommended Updates for mkinitrd (http://support.novell.com/techcenter/psdb/24c7dfbc3e0c183970b70c1c0b3a6d7d.html) at the Novell Technical Support Center.

A.2.3 Managing Multipathing for Devices and Software RAIDs

The following changes were made to this section:

Location	Change
Section 5.1.3, “Guidelines for Multipathing,” on page 42	Effective in SUSE Linux Enterprise Server 10, a root partition on multipath is supported only if the <code>/boot</code> partition is on a separate, non-multipathed partition. Otherwise, no bootloader is written.

Location	Change
Section 5.1.5, “Device Mapper Multipath I/O Module,” on page 43	Changes were made for clarification.
Section 5.1.4, “Device Mapper,” on page 42	DEVICES should be DEVICE.
Section 5.2.3, “Configuring mdadm.conf and lvm.conf to Scan Devices by UUID,” on page 45	DEVICES should be DEVICE.
Section 5.6, “Configuring Multipath I/O for the Root Device,” on page 47	This section is new.

A.2.4 Managing Software RAIDs

The following changes were made to this section:

Location	Change
Section 6.1.8, “Multi-Disk Plug-In for EVMS,” on page 59	Technical corrections were made.
Section 6.1.9, “Device Mapper Plug-In for EVMS,” on page 59	Technical corrections were made.
Section 6.5, “Managing Disk Failure and RAID Recovery,” on page 65	This section was expanded for clarification.

A.2.5 Managing Software RAIDs with mdadm

This section is new.